Economic Model Predictive Control of Chemical Processes

by

Omar Santander

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Chemical Engineering

Waterloo, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The objective of any chemical process is to transform raw materials into more valuable products subject to not only physical and environmental but also economic and safety constraints.

To meet all these constraints in the presence of disturbances the processes must be controlled. Although nowadays there are many available control techniques available Model Predictive Control (MPC) is widely used in industry due to its many advantages such as optimal handling of interactions in multivariable systems and process constraints.

Generally, the MPC strategy is implemented within a hierarchical structure, where it receives set points or targets from the Real Time Optimization (RTO) layer and then maintains the process at these targets by calculating optimal control moves. However, often the set point from the RTO may not be the best optimal operation or it may not be reachable thus motivating the integration of the RTO and MPC calculations into one single computation layer.

This work focuses on this idea of integrating RTO and MPC into one single optimization problem thus resulting in an approach referred in literature as Economic Model Predictive Control (EMPC). The term “Economic” is used to reflect that the objective function used for optimization includes an economic objective generally used in RTO calculations. In this thesis, we propose an EMPC algorithm which calculates manipulated variables values to optimize an objective consisting of a combination of a steady state and a dynamic economic cost. A weight factor is used to balance the contributions of each of these two terms. Also, the cost is defined such as when the best economic steady state is reached the objective is only influenced by the dynamic economic cost.

An additional feature of the proposed algorithm is that the asymptotic stability is satisfied online by enforcing four especial constraints within the optimization

problem: 1-positive definiteness of the matrix P defining the Lyapunov function, 2-contraction of the Lyapunov function with respect to set point changes, 3-contraction of the matrix P with respect to time and 4- Lyapunov stability condition. The last constraint both ensures decreasing of the Lyapunov function and also accounts for the robustness of the algorithm with respect to model error (uncertainty).

A particular novelty of this algorithm is that it constantly calculates a best set point with respect to which stability is ensured by the aforementioned constraints. In contrast to other algorithms reported in the literature, the proposed algorithm does not require terminal constraints or terms in the cost that penalize deviations from fixed set points that often lead to conservative closed loop performance.

To account for unmeasured disturbances entering the process, changes in parameters are also explored and the algorithm is devised to compensate for these changes through parameter updating. Accordingly, the parameters are included as additional decision variables within the optimization problem without the need for an external observer. The stability of the parameters estimation is ensured through the set point constraint mentioned above.

To demonstrate the capabilities of the proposed algorithm, it is tested on two case studies: a simpler one involving a system of 4 nonlinear ODEs describing an isothermal nonlinear reactor and a larger problem involving a non-isothermal Williams-Otto reactor with parallel reactions. The dynamics of the latter reactor consists of a set of nonlinear ODE describing the evolution of the process temperature and concentration of the different species.

The simulations for the isothermal reactor showed that the proposed algorithm not only outperformed (in terms of an economic function) alternative formulations, but addressed all their limitations. In addition, when there was a parameter modification, this was adapted in a finite time. In terms of the non-isothermal reactor, the simulations demonstrated that not only the best steady state could be computed, but also the states were steered to it satisfying the online stability property.

Acknowledgments

First of all, I would like to thank my supervisors, Professor Hector Budman and Ali Elkamel. Thank you for this great opportunity, I deeply appreciate all the knowledge that you shared with me, your advices, your patience and all the help that you have provided me all this time.

In the second place, I would like to thank my colleges and members of Hector's research team, Rubin and Yuncheng who gave me some feedback about my research or helped me with some course inquiry and whom I spent a very good time. I would also like to thank my Mexican friends Ricardo and Memo whom I liked to party with. Without you guys Waterloo would have been boring.

Thirdly, I would like to thank CONACYT and SEP for all the financial support that have given to me during my Master studies.

Finally, I would like to thank UW-Chemical Engineering staff for all the technical and/or documentation help that have given to me.

To my mother, sisters and family. Without your support I would not be here.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Chemical Plants are designed with the task of transforming raw materials into more valuable products. These transformations must occur in the most efficient way so as to attain different goals such as the yield need to be highest possible, the amount of contaminants or by-products has to be minimal, the energy employed in the process has to be minimum etc. Furthermore, these transformations have to be carried out under economical, physical and environmental constraints and they must be robust to process disturbances such as variations in process settings such as temperature, input flows, pressure etc., or variations in raw material quality. To achieve these goals advanced control structures are a must since they can not only deal with MIMO systems, but incorporate optimization tools that consider directly process constraints.

During the past decades several control strategies have been developed and applied ranging from single input single output controllers to multivariable decouplers and model predictive controllers (MPCs). Now more than ever the chemical industry worldwide is witnessing such levels of competition that require processes to be as profitable as possible.

The conventional hierarchical control structure implemented in most process industries, shown schematically in Figure 1.1, involves an RTO (real time optimization) level on top of a multivariate control level realized by an MPC or other multivariable strategies followed by lower level single input single output controllers (PIDs). The RTO is generally executed to maximize a steady state economic cost with respect to optimal controlled and/or manipulated variables values that are then used as set points in the lower level multivariable control strategy. Thus, the RTO levels computes the targets and the multivariable controller (e.g. MPC) control the system around these targets. Then, the multivariable controller strategy calculates either directly the values of the manipulated variables or indirectly the set points for low level single input single output controllers (e.g. PIDs) regulating each one of the manipulated variables of the process. Although this hierarchical strategy has resulted in good performance there is a great opportunity for improvement since chemical processes are rarely at steady state and thus the steady state set

points calculated by the RTO and enforced by the MPC controller may not be optimal in transient scenarios.

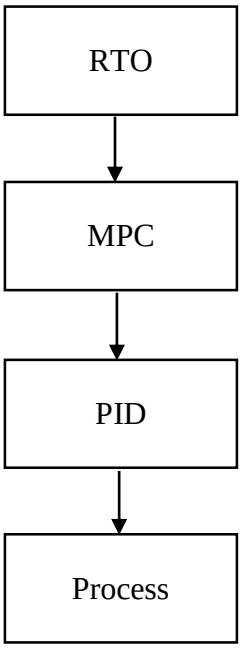


Fig. 1.1: Hierarchical structure in process operations.

1.1 REAL TIME OPTIMIZATION

RTO is the first component (upper layer) of the actual hierarchical control structure shown in Fig 1.1. It usually accounts for market, process and corporative information (raw material cost, inventory etc.), and consequently it is often described by a complex nonlinear algebraic model.

Since the RTO calculation maximizes steady state profitability, it is commonly executed only when the system is detected to be close to a steady state. Therefore, the RTO sampling period is much larger than the sampling period used by the MPC controller that performs dynamic regulation. In a typical chemical process the sampling period of RTO will be in the order of days or several hours whereas MPC will use sampling periods in the order of minutes.

The algorithm for the RTO may be summarized into these four steps:

i) Evaluate the data and detect new steady state

ii) Validate and reconcile the data

iii) Estimate model parameters and update the model

iv) Using optimization and the nonlinear algebraic model describing the process steady states obtain a new steady state

The optimization is usually subject to constraints and the cost function involves a steady state economic index.

Once a new steady state is calculated the values of the prospective controlled and/or manipulated variables at the new steady state are sent and used as set point by MPC or any other multivariable control strategy used as shown in Figure 1.1.

1.2 MODEL PREDICTIVE CONTROL

MPC performs an online optimization over a finite horizon with a sampling period of the order of minutes.

The conventional cost function, also referred as the stage cost, is usually quadratic involving the sum of squares of the deviations of the states and/or manipulated variables from the steady states and steady state inputs computed by the RTO layer. Usually, weight factors (matrices) are assigned to the different terms in the stage cost so as to tune the performance of the system.

MPC employs the use of a dynamic model and uses a prediction of the states based on this.

The general procedure of MPC is schematically represented in Figure 1.2 and proceeds according to the following steps:

1. Measure the actual state
2. Compute de manipulated variables through optimization that will minimize the cost subject to the system constraints
3. Apply the first component of the solution vector (optimal input vector)
4. Repeat again the procedure until the desired set point has been reached

MPC not only is applied to reach a particular set point but it is also used to maintain this set point in the presence of process disturbances. By using a model of the process MPC can enforce process constraints over a prediction time horizon and compensate for interactions occurring in multiple input-multiple output systems (MIMO). The mathematics of MPC is described elsewhere and therefore it is not presented here for brevity.

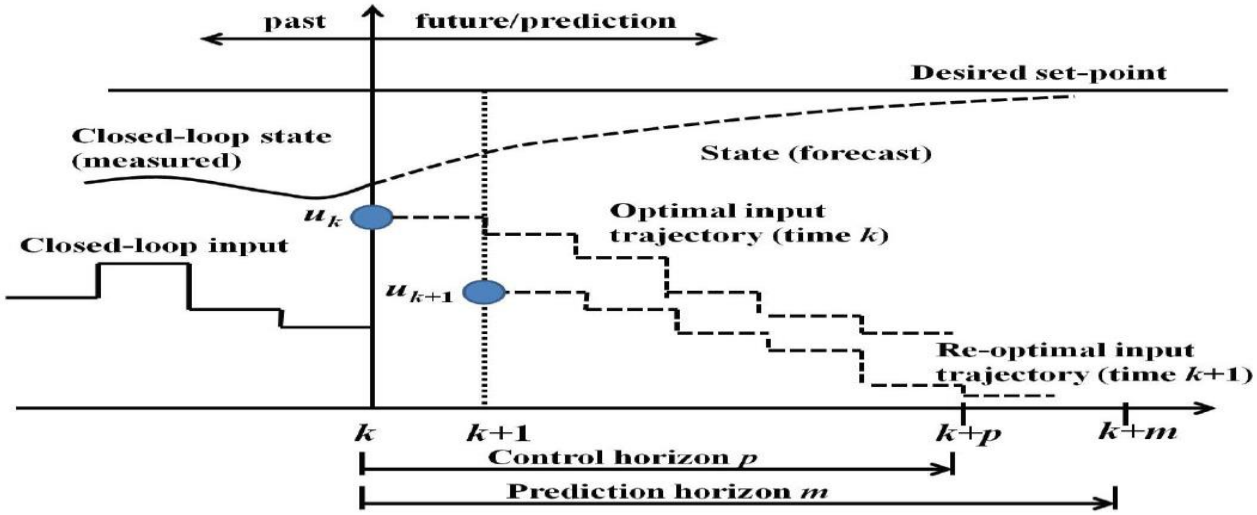


Fig 1.2: MPC descriptive diagram (Mahmoud 2012).

1.3 ECONOMIC MODEL PREDICTIVE CONTROL

The motivation for EMPC as an alternative to the hierarchical control structure described in Figure 1.1 is that the latter is not optimal since it relies on the basic assumption that the system can instantaneously achieved the optimal steady state calculated by the RTO while transients are effectively ignored. Moreover, due to dynamic disturbances continuously occurring, the process may never be at steady state thus invalidating the basic assumption for optimality of the RTO calculations. Figure 1.3 illustrates the case where the line projection of all the steady state solutions is not optimal. Figure 1.3 describes the profit as a function of the state and an input assuming for simplicity a one state-one input system. The figure is based on deviation with respect to the economic global optimum. It is clear that the best steady state, defined as the most profitable one, is not the optimal state. Therefore, so as to operate around the economic global optimum, the system must be controlled dynamically. Furthermore, due to the nonlinear nature of chemical processes, regimes may exist that outperform the best steady state such as periodic regimes.

EMPC still maintains many of the strengths of MPC, like for instance, the use of dynamic MIMO models, the direct handle of constraints, online computation etc. However, in contrast with conventional MPC, it uses directly a cost function that involves an economic performance index instead of a quadratic stage cost. This means that it does not rely on any particular set point value thus eliminating the need for the RTO level. The price for by-passing the RTO level is that stability has to be assessed with respect to a potentially time varying set-point. Enforcing stability is particularly challenging since now we are dealing not only with a nonlinear model as in regular NMPC (Nonlinear MPC) but also with a generally nonconvex stage costs instead of a quadratic cost. Nonconvex stage cost might also imply the existence of multiple minima.

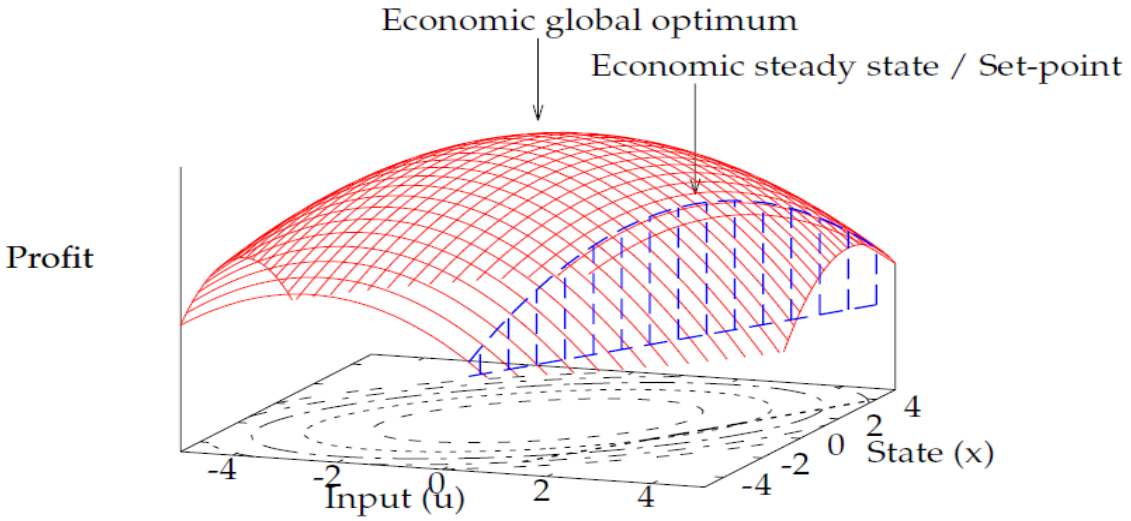


Fig. 1.3: Graphic showing that the best set point is not the most profitable one (Rawlings 2012).

1.4 OBJECTIVES OF THE RESEARCH

The main objectives accomplished during the study were as follows:

- Formulate a novel EMPC algorithm that uses dynamic and steady state models which is both stable and robust to model errors
- The performance of the proposed algorithm is enhanced by avoiding terms that limit closed loop performance such as terminal constraints and terms that penalize deviations with respect to fixed set-points
- The proposed algorithm was compared and found to be superior to other reported EMPC formulations
- The proposed algorithm is designed to account for parameter changes
- The performance of the proposed algorithm is illustrated with two practical case studies

1.5 OVERVIEW

Chapter 2.

It contains all the literature review and antecedents that were used for the research.

Chapter 3.

In this chapter we propose our new EMPC algorithm and we demonstrate its robustness and stability. We also state all our assumptions and modifications. Simple simulation examples are provided and the performance of our approach is compared to similar formulations.

Chapter 4.

In this chapter we extend the implementation of the algorithm to a larger and more realistic example than the one used in Chapter 3. The problem of parameter estimation is also considered in this Chapter. In addition, conditions for the feasibility of the algorithm is also developed.

Chapter 5.

This chapter provides conclusions, remarks and future work.

Appendix A.

The code for our EMPC is described. Appropriate modifications in the subroutines need to be implemented so as to tailor the EMPC to a particular problem. Pertinent instructions are given for the execution of all the simulations shown in this work.

Chapter 2

LITERATURE REVIEW

2.1 INTRODUCTION

The research about both MPC and RTO is fairly extensive and it has been reported in both journal articles and books (Camacho and Bordons, 2004; Grune and Pannek, 2011; Rawlings and Mayne,2009). In terms of integration of RTO and MPC, many books have described the hierarchical structure briefly described in the previous chapter (Marlin, 1995; Luyben and Tyreus,1999).

As it was previously pointed out, the hierarchical structure that combines a separate RTO and MPC levels has some disadvantages as follows (Ellis, Durand, Christofidies, 2014):

- Discrepancy in the models used for RTO and MPC may result in some steady states that are unreachable by the MPC layer
- Since the RTO layer is executed infrequently a change in optimal steady state due to disturbances may not be calculated in time thus affecting the overall economic performance
- Set points corresponding to steady state operation might not be the most profitable for dynamic situations and sometimes it is more profitable to operate dynamic set points

Consequently, a different approach is needed and using an economic performance criterion directly as the cost function of the MPC controller may be one way to overcome these drawbacks.

In the literature, there are some EMPC formulations that tackle many different problems (infinite horizon, periodic operation, terminal region, pointwise terminal constraint etc.). This thesis is based mainly on the theory developed by Rawlings and workers that were the first to study the

idea of combining an economic index together with a stage cost that penalizes deviations from steady states values of states and inputs (Christofidies et al. 2014).

For completeness, before reviewing Rawlings work, we will review some other important approaches.

2.2 ONE LAYER APPROACH

Adetola and Guay (2010) proposed a single layer strategy involving the integration of RTO and MPC into one optimization problem. The algorithm seeks for the best operating condition while also adapting parameters to cope with disturbances.

Combining an extremum seeking set point design, a barrier function to account for constraints, a pre-defined Lyapunov function and a finite time parameter estimator, they formulated an update law for a time-varying set point trajectory. The algorithm has guaranteed Lyapunov stability with respect to this set point.

Once the update law for the set point was calculated, an MPC approach was implemented to track after this set point while ensuring that the deviations of the states from the set point at a terminal time were bounded.

The resulting approach was a min-max problem where maximization was done with respect to bounds on the parameters so as to provide robustness (worst case scenario) and the minimization was done with respect to the control actions. The stage cost consisted of a tracking term with a penalty related to the terminal set constraint. The parameter update law and the reference trajectories were used as constraints within the optimization problem. The parameter uncertainty set was also updated in this formulation to reduce conservatism. This one layer approach has several tuning parameters and extensive simulation must be performed in order to get good performance. In contrast, in the current research our proposed approach eliminates some of the limitations of the algorithm reviewed in this section as explained later in the thesis.

2.3 LYAPUNOV MPC ALGORITHM (LMPC)

Christofidies et al. (2012) proposed an EMPC algorithm based on the Lyapunov stability concept and a time unvarying economic cost. Their formulation is referred to as LEMPC (Lyapunov Economic Model Predictive Control). They considered a dynamic system under disturbances for which they proposed Lyapunov controllers which rendered the origin of the nominal closed loop system asymptotically stable for any state inside a stability region (ρ).The controller was subject to constraints in the inputs, and the Lyapunov function was forced to satisfy various constraints including upper and lower bounds, monotonic decrease of the time derivative and boundedness of the derivative with respect to the states.

This work included two different approaches based on the type of data available for feedback: synchronous measurements versus asynchronous and delayed measurements. Moreover, a sub-stability region (ρe) inside ρ was considered so as to ensure that under bounded perturbations the states would always remain inside ρ. Depending on the location of the states with respect to either the ρe or ρ regions and time, two modes of operation were developed to ensure boundedness:

i- if the states were inside ρe, the system was allowed to evolve freely provided that time<t', where t' was a pre-specified time by the user and that the system would remain inside ρe and ii-If the system was outside ρe but inside ρ the states would be driven inside ρe for time<t', otherwise, the states would be pushed to the origin.

The modes of operation were used as constraints in the optimization problem which consisted on minimizing a time-invariant economic cost.

Both feasibility and stability were proven using Lyapunov functions' arguments.

For the EMPC with asynchronous measurements, an auxiliary time-varying variable was introduced to account for the delay.

For the implementation, the current state was estimated based on the delayed measurement, the nominal dynamic model and the optimal inputs calculated at the previous time interval. Once it was estimated and based on its location (inside or outside ρe) and the chosen time (<t') the mode of operation was chosen. The task of estimating the states was accomplished by introducing an equivalent related constraint into the formulation.

Ellis and Christofidies (2013) developed an LEMPC with time varying stage cost. In this case, the economic cost time variation was of similar order of magnitude as the process dynamics. As explained above (Christofidies et al. 2012) stability was guaranteed by using a Lyapunov stable controller that drove the states to a fixed origin.

Since due to the varying nature of the stage cost, there were multiple steady states, several stability regions were correspondingly considered. Thus, for each steady state (xs) a stability region (ρs) was found. Then, a union of stability regions referred to as the *union set* X_U was considered for analysis. The union set was approximated by analytic mathematical expressions obtained through curve fitting or convex optimization.

The process for generating each stability region was as follows:

- Select a steady state from the *union set* and divide the state space around xs into several discrete points.
- For each one of these points, check whether the time derivative of the Lyapunov function is decreasing along the trajectory when the Lyapunov controller is applied
- If the time derivative is not decreasing for a given point, set the Lyapunov function value equal to ρs

Once all stability regions were generated, the union set X_U could be computed.

Since disturbances were considered, a subset of the union set X_U had to be generated, referred to as $X_U^{'}$. $X_U^{'}$ was the largest subset for which the state trajectory remain inside X_U in the presence of disturbances over one sampling period. Clearly the size of $X_U^{'}$' depended on the bounded disturbances' magnitudes and on system dynamics.

Two operation modes were considered: mode 1 ensured that if the system initially (measurement) was inside $X_U^{'}$, it would evolve freely inside this region and mode 2 where if the state was outside $X_U^{'}$ but inside X_U, the states were driven into the region $X_U^{'}$. Note that in comparison to the previous work by the same group reviewed at the beginning of this section (Christofidies et al. 2012), time did not play a role in this case in defining the operation modes. The two operation modes were implemented in the LEMPC formulation through the constraints.

More recently, Ellis and Christofidies (2014) proposed a method to monitor the performance of EMPC. The theory regarding EMC and stability was similar to (Christofidies et al. 2012), and (Ellis and Christofidies 2013). Thus, a Lyapunov controller was used that rendered the origin of the dynamic system asymptotically stable. As in their previous studies, stability regions and sub stability regions were considered, and also the strategy was implemented using two modes of operations. The formulation of the LMPC was identical to the one described in (Ellis and Christofidies 2013).

Since the economic cost was time varying, its distribution was not normal and it was auto correlated with time and consequently it was not a simple metric for monitoring the performance. Instead, a residual variable was defined as the economic cost minus the economic cost using the nominal system. The economic cost of the nominal system was calculated with past data that was filtered using an exponentially weighted moving average (EWMA) filter.

Upper and Lower bounds were defined during the time period Δt. These bounds were defined as functions of mean and standard deviation of past data. If the output from the EWMA filter was found to exceed one of these bounds during the time interval Δt, poor performance was detected thus prompting for the need to improve the controller.

Ellis and Christofidies (2014) proposed a two layer approach, where in the upper layer an EMPC was implemented and the calculated optimal trajectory was then sent to a lower control layer consisting of an LMPC which objective was to track the trajectory. Finally, the optimal control actions from the LMPC layer were sent to the process.

The reference trajectory (xr) was assumed to be in a compact set "B" of reference trajectories. Moreover, the variation with time of the reference trajectory was sufficiently bounded so as to allow tracking accuracy. An additional constraint was added that dealt with the bound of the variation of the Lyapunov function with respect to xr.

As in the previous studies with Lyapunov controller based formulations, a stability region (ρxr) was computed for each trajectory in B and then the total stability region (ρ) was derived from the union of all these stability regions. The computation of each ρxr was similar to the one described in (Ellis and Christofidies 2014).

An EMPC was formulated as a minimization problem that was subject to a nominal dynamic model, constraints in the inputs, a constraint on the reference trajectory xs that forced it to remain inside B and a constraint in the time variation of xs.

The calculated reference xs was sent to an LMPC which was formulated as a minimization problem where the stage cost was a tracking term. One of its constraints was a Lyapunov stability condition.

The stability and robustness properties of this joint formulation (EMPC + LMPC) were enforced through the constraints of the LMPC algorithm.

2.4 ECONOMIC MPC

As it was pointed out in the Chapter 1, the current research is based on the work of Rawlings and workers. They reported a series of studies regarding EMPC which are briefly summarized in this section.

Rawlings et al. (2009) proposed a receding horizon control algorithm for nonlinear plants with stage cost that is not necessarily convex. The authors demonstrated that if the best steady state

was used as terminal point wise constraint and for feasible initial conditions, the average economic performance of their algorithm was at least as good as the best steady state. No explicit assumption was made in that study regarding stability.

Then, the authors investigated the use of a periodic terminal constraint instead of a terminal best steady state constraint. The periodic trajectory used as terminal constraint was derived from a minimization problem where the initial state had to be the same as the final one with a fixed period Q. It was shown for this case that the steady state solution, calculated by the minimization of the stage cost subject to the steady state model, was a special case of the periodic constraint based case. In general, it was found that the economic cost with the periodic constraint was superior to the cost obtained at the best steady state. Moreover, the authors showed that if the period Q1 was a multiple of some other period Q2, the resulting cost using Q1 was not worse than the cost obtained when using Q2. Also, it was shown that the periodic terminal constraint produced a periodic control law. Finally, it was proven that the closed loop system had an average economic performance which was as good as with the optimal periodic solution Q.

Thirdly, the use of average constraints was explained and implemented. The average constraints were satisfied asymptotically in the receding horizon control strategy. An EMPC was tested subject to average constraints. It was demonstrated for this algorithm that if the initial condition was feasible, then feasibility was ensured for all subsequent times and the average constraints were also met.

Finally, a section for systems which are optimally operated at steady state was discussed. It was stressed that a system was considered to be optimally operated at steady state if the average performance was lower bounded by the best steady state stage cost. Then it was proven that linear systems are optimally operated at steady state for any arbitrary convex stage cost.

A key contribution of the work of Rawlings and co-workers is the derivation of relatively simple stability conditions for EMPC based on Lyapunov arguments (Rawlings et al. 2011).

Some assumptions were made in this work to prove stability.

First, weak controllability was assumed whereby the input cost needed to steer the state to the best steady state was not very large. Secondly and most importantly, the system was considered to be strongly dual. This assumption implied that there exist Lagrange multipliers which made the steady state the unique minimizer of a minimization problem in which the stage cost was a modification of the original cost referred to as *rotated cost*. It was demonstrated that the *rotated cost* was a Lyapunov function which was decreasing monotonically and it was both upper and lower bounded. Hence, it was possible to conclude that the steady state of the closed loop system was asymptotically stable.

The algorithm was applied to a CSTR where the closed loop model was made strongly dual by adding tracking terms with weights in the economic stage cost.

Strong duality had to be checked numerically by solving off-line the minimization of the steady state problem with the tracking terms and the minimization with multipliers and by testing whether the solutions were identical. Once that identical solutions were computed, the weights for the tracking terms were directly obtained from the diagonal elements of the Hessian matrix.

Based on a predefined Lyapunov function, Rawlings et al. (2012) developed a formulation for their EMPC algorithm for a continuous system f(x,u) and continuous stage cost l(x,u) with terminal pointwise constraints.

Weak controllability was assumed. Then, asymptotic average performance was demonstrated where for a feasible initial condition the closed loop system has an asymptotic economic performance that was not worse than the best economic steady state.

Secondly, the stability was treated by using the concept of dissipativity. In this case, in contrast with their previous study (Rawlings et al. 2011) the strong duality condition was relaxed so as to make the closed loop system dissipative.

Dissipativity was tested by a suitable inequality for all the states and manipulated variables within the feasible region. The left hand side of this inequality was the difference between the storage function at time k and k+1 and the right hand side involved the supply rate.

Strict dissipativity was accomplished by adding a positive definite term that was defined with respect to the steady state in the left hand side of the dissipativity inequality.

Then, stability through dissipativity was demonstrated by first proving that the minimization problem with the rotated cost gives the same solution as the minimization problem with the economic cost. Then, using dissipativity and the rotated cost as a Lyapunov function, it was shown that the Lyapunov rate was strictly negative.

Furthermore, an example was shown where using the dissipativity concept was an effective alternative for relaxing the strong duality property which for this example could not be satisfied.

In addition, one method for making the steady state the best operation condition was explained. This method consisted in modifying the economic stage cost by adding a function that could satisfy strict dissipativity thus providing asymptotic stability around the best steady state. The proposed function depended on the deviations from the best steady state and it was usually quadratic. This function was used to provide convexity to the original economic cost while the best economic steady state was the optimal solution for this modified cost.

Some extensions of the EMPC were developed to deal with non-steady states and averages of states' formulations.

In terms of non-steady states' formulations a periodic terminal constraint was introduced. This constraint or trajectory was computed offline through a minimization problem subject to the condition that the initial state must be the same as the final. This constraint effectively resulted in a time varying feedback law. It was also proven that the average economic performance of this formulation was not worse than the best periodic economic solution.

For the case of averages of states' an EMPC was constructed through a model update equation that was introduced as a constraint.

Finally, some notions for optimal operation at steady state with and without average constraints were proposed. In the case of average constraints, it was proven through strict dissipativity that the best regime for economic operation was the best economic steady state.

Rawlings et al. (2011) also proposed EMPC with terminal cost instead of terminal pointwise constraints. In this EMPC approach, the stage cost consisted of an economic function and a penalty terminal function related to the terminal constraint. The terminal constraint enforced the state to end up within a terminal region (XF) instead of a terminal point. Through several assumptions related to the constraint compact set, the economic cost, the terminal function, the dynamic system, the storage function, the Lyapunov function in the terminal region, and the concept of strict dissipativity some properties were demonstrated for this formulation.

Different costs were introduced for the analysis including the concept of rotated cost mentioned above, a shifted stage cost that was equal to the economic cost minus the economic cost evaluated at steady state, a modified terminal cost and a rotated regulated cost function which is equal to the sum of the rotated cost over the horizon plus modified terminal cost. Using the regulated cost function both a standard and an auxiliary EMPC were defined.

Secondly, several lemmas related to the rotated stage cost were developed using the assumptions previously stated. For example the modified terminal cost was shown to be decreasing towards the terminal region and to be positive definite with respect to the steady state. Then, it was shown that the standard and auxiliary formulations have the same solution. After this, it was proven that if the system was dissipative and using the auxiliary minimization problem it was shown that the steady state was an asymptotic equilibrium point. Finally, defining averages, it was demonstrated that the average performance of the closed loop system was at least as good as the best steady state.

Thirdly, a candidate terminal cost function that could satisfy the stability assumption was proposed. In the first place, additional assumptions were pointed out: the dynamic system and the stage cost were twice continuously differentiable, the linearized system is stabilizable and the dynamic system belongs to an invariant set.

Then, candidate terminal cost functions were proposed based on the availability of a storage cost function. For the case of an unknown storage cost a quadratic Lyapunov terminal cost function was proposed and the state space region where it could meet the stability assumption through the use of a linear controller was defined. For the case of a known storage function, the concept of dissipativity was used and stability for a particular terminal cost function was achieved by the use of a linear controller.

In Rawlings et al. (2011) a new method was proposed for achieving asymptotic convergence to a desired set point using average constraints. Compare to their previous approaches, the system did not need to satisfy any strong duality or dissipativity properties. The EMPC formulation was subject to not only point wise constraints but an average constraint which was satisfied asymptotically and that was updated with time.

In order to demonstrate the convergence using average constraints, some definitions and lemmas had to be proven. First of all, a sequence that was essentially converging to zero was defined. Then it was demonstrated that a sequence with zero even moments is also converging to zero.

Finally, using an update equation recursively for the set of the average constraints and the auxiliary output function that was used in the average constraint as a tracking term, and given a feasible initial condition, it was demonstrated that this EMPC formulation was feasible and asymptotically stable towards the steady state.

It was also demonstrated that if an average constraint is added along with the update equation, the system is steered asymptotically towards the desired steady state.

In this approach dissipativity did not have to be considered, consequently, there was no need to modify the stage cost as in previous formulations.

Biegler et al. (2013) developed numerical methods to solve the EMPC formulations previously discussed. In this paper, MPC was considered for systems modeled by both differential and algebraic equations (DAE). The continuous model was discretized and the constraints were enforced at the boundary of each discrete element The MPC formulation consisted of either terminal pointwise, terminal region and/or terminal cost constraints.

There are various methods that can be employed to solve systems that are modeled by DAE. More information about these methods can be found in (Amrit 2011, Biegler 2010)

In Rawlings MPC formulation, a direct simultaneous method was preferred due to some numerical advantages such as:

- It does not use any intermediate solution, reducing the computational effort.
- It is robust for problems that are ill-conditioned.
- It allows direct enforcement of state and input variables that are discretized with the same level of discretization as the states.

In general, the direct methods deal with the transformation of the infinite continuous problem (EMPC) into a very large finite nonlinear problem that can be solved using NLP solvers. For the simultaneous approach both the manipulated and states are discretized as reported in (Amrit 2011, Biegler 2010).

The formulation of the simultaneous approach involves approximating the algebraic and differential states and input profiles by a family of polynomials. For the case of differential states, monomial basis representations were used along with an equation that enforced continuity. This basis representation was function of the first derivative of the polynomial evaluated at a colocation point.

For the inputs and algebraic states, the profiles were approximated by a Lagrange basis representation, i.e. Lagrange polynomial. Besides, a zero order hold (ZOH) delay was enforced by adding an additional constraint that fixed the input profile within each element along each time interval.

Following the above operations, the MPC or EMPC formulations were transformed into NLPs that could then be solved using special solvers (IPOPT). Since the information for second and first order derivative was crucial, automatic differentiation was also employed. Automatic differentiation provides information about the value of derivatives without any truncation error. Two examples involving an evaporator and a reactor were simulated for which EMPC and MPC were solved as NLPs using IPOPT and automatic differentiation.

Finally, in Rawlings et al. (2012) a summary of their ideas including the use of a stage cost as a Lyapunov function, terminal cost, terminal pointwise constraint, average quantities, enforce convergence and periodic operation was provided. This paper also summarized the work done in (Amrit 2011).

2.5 LARGE SCALE APPLICATIONS OF EMPC

There are several papers that deal with large scale applications of EMPC. For brevity we will summarize in this section only a few contributions.

Gopalakrishnan and Biegler (2013) formulated an EMPC that could reduce the operating cost of a pipeline network. It was stressed that the best operation condition was a dynamic one in view that the steady state based operation neither considered particular dynamic operating conditions such as peak loads.

The periodic EMPC formulation used was very similar to the work performed by Rawlings with a periodic terminal point wise constraint, and the stage cost was modified in order to make it convex by adding tracking terms.

The dynamic model consisted of a typical natural gas network with suppliers, consumers, intermediate nodes and compressor stations. The nodes were linked through pipes. The flows and pressures satisfy the continuity equation and pressure drops in the pipes were also considered. Network equations were then used to link between flows and pressures values in each node.

The EMPC problem was formulated as an NMPC with constraints including terminal values, pressure limits, contract pressure etc. and the stage cost was equal to the total compression energy. Some tracking terms were added to the stage cost to make the system asymptotically stable towards the optimal periodic trajectory. The NLP resulted in more than one hundred thousand equations and variables.

It was shown in this study that EMPC resulted in better performance as compared to the standard MPC, not only for the nominal case but also for the case with measured disturbances.

Biegler et al. (2011) proposed an EMPC formulation for an industrial size air separator which consisted of two integrated cryogenic units. The ASU (Air Separation Unit) was modeled using tray by tray energy and mass equations. This model had over three hundred ODEs. The stage cost was equal to the utility cost that was function of the electricity price and also included tracking quadratic terms.

In the EMPC formulation, a periodic constraint was enforced. In contrast with previous formulations, it was not necessary in this case to perform an offline computation of the periodic trajectory. Instead, the system converged to the optimal periodic trajectory based on optimality of the proposed cost. Asymptotic stability of this algorithm was proven based on Lyapunov arguments.

This paper also dealt with an infinity horizon formulation in which a discount factor in the stage cost was considered. The discount factor converted the future profit into a present cost. In this case, a non-periodic constraint was used but the tracking terms in the stage cost were used to enforce periodicity.

Biegler et al. (2012) also proposed an infinite horizon Robust EMPC for the ASU. The EMPC was formulated as an infinite NLP subject to standard constraints such as constraints in the states, inputs, and on the dynamic model but in this case the stage cost did not include any discount factor. Using a similar analysis as in (Biegler et al. 2012) the asymptotic stability to the origin was demonstrated.

Using the concept of rotated cost, it was proven that an economic stage cost could be sufficiently transformed by adding tracking terms so as to satisfy asymptotic stability.

An alternative approach was proposed by using Lagrange functions of the nominal system and the best periodic operation, which was repeated over and over again at the end of the infinite horizon, and by defining a Lyapunov function as the subtraction of the nominal and the system with the terminal periodic operation.

Finally, robustness was analyzed based on input to State stability analysis, where an uncertainty term in the dynamic model was considered.

Qin et al. (2011) applied an EMPC for climate control of a commercial building in Milwaukee, WI, USA. The main targets were to reduce the overall energy consumption and lower the peak power demand. Indeed, it was pointed out that EMPC was particularly suitable for building pre-cooling optimization since steady state was barely reached during the pre-cooling phase.

In the proposed EMPC, the stage cost was a function of the energy cost, average consumption, peak demand cost and peak power demand. The minimization of the stage cost was formulated as a min-max problem, where the stage cost was minimized but a term within the stage cost related to the peak demand was maximized. The min-max problem was converted to a linear program subject to standard constraints.

The construction of the inequality constraints were based on the peak cost demand whereas the equality constraints came from the dynamic model. Upper and lower bounds were used for the power and temperature.

To model the dynamics of the system and to accommodate the thermal distribution, each floor of the building was divided in several zones. The dynamic model was an auto regressive exogenous model (ARX) identified either from simulated data or from actual process data. To simulate the process, the Energy Plus software was used. In the dynamic model, the power and temperature were functions of the set point temperature.

EMPC was shown to result in great savings, perform better than previously proposed controllers for this system and also outperformed open loop strategies.

For real application of the algorithm, one floor in an office building in Milwaukee was selected. The floor was divided in 25 climate controlled zones. For system identification, a PRBS (pseudo random binary sequences) was used.

Following the implementation of EMPC it was found that the power consumed during peak hours was less as compared to the baseline scenario (building operating schedule)

Finally, Idris and Engell (2012) studied the economics of a nonlinear MPC for a catalytic distillation column. The system was modeled by discretized DAEs. This study compared a purely economic cost based MPC to an economic based MPC with the addition of a tracking cost and to an MPC solely based on a tracking cost.

The tracking cost was the square of the deviations between the states and inputs and their corresponding steady state values. The minimization was subject to standard constraints (upper and lower bounds, DAE, inequality etc.). To compensate for plant model mismatch, a bias correction equation was used.

The catalytic distillation column (CD) was of packed bed type with a reboiler and a condenser. The column consisted of an enriching packing segment, two catalytic segments, and a stripping packing segment. The esterification chemical reaction took place in the catalytic segments. The CD was described by an equilibrium stage approach using MESH equations. The CD consisted of fourteen trays with ninety ODEs and over five hundred algebraic equations.

In order to speed up the computations, the numerical calculations were done in two parallel sections where one section computed the phase equilibrium and the other the properties. The CD was described by two different models, a full model that represented the true plant and a reduced model that was used by the controller.

Two control structures were considered. The outputs were the same for both but one structure had more inputs than the second one. The economic cost consisted of the profit of the operation and for the case that a purely economic cost was considered, the outputs were used as a quality constraints. For the case that the cost combined an economic cost term and a tracking term, the latter was equal to the square of the deviations between outputs and their corresponding set points.

The SNOPT software was used to solve the optimization problems. For the case of a purely economic cost, its performance outperformed the others approaches (tracking and economic plus tracking) but the computational requirements were very large compare to the other approaches considered in this work.

Chapter 3

Formulations, Stability and Case Studies

3.1 INTRODUCTION

In this chapter the proposed new EMPC algorithm is introduced including the stability conditions and a simple case study to illustrate the method. The proposed EMPC is stated as a dynamic optimization problem that is based on a receding horizon formulation subject to constraints in manipulated variables and additional specific constraints to enforce stability. The stage cost is not restricted to be convex and it is solely determined by the economic objectives of the process. Additionally, the control prediction horizon is finite. In 3.2, the notation and the formulation of the algorithm as an optimization problem are explained. In addition, the specific constraints used to enforce closed loop stability are explained in detail. Simulation studies of our EMPC applied to a reactor unit with parallel reactions are shown in 3.3.

3.2 EMPC ALGORITHM

The development of our algorithm is explained next

3.2.1 Notation and Formulation

Bold symbols represent vectors and/or matrices, whereas non-bolded characters represent scalars. The Euclidean norm of a vector is represented by $|*|$.

We begin this section by making some definitions and assumptions.

Definition 3.1 (Positive definite matrix)

A square matrix **P** is positive definite if:

$$\boldsymbol{P} = \boldsymbol{P}^T$$
$$\boldsymbol{P} > 0$$

Definition 3.2(Positive definite function)

A function $\rho()$ is positive definite with respect to $\boldsymbol{x} = \boldsymbol{a}$ if:

$$It\ is\ continuous, \rho(\boldsymbol{a}) = 0\ \ and\ \rho(\boldsymbol{x}) > 0 \quad for\ all\ \boldsymbol{x} \neq \boldsymbol{a}$$

Definition 3.3 (Class $\mathcal{K}$ function).

A function $\Upsilon(): \mathbb{R} \longrightarrow \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}$ if it is continuous, zero at the origin and strictly increasing.

Lemma 3.1

Given a positive definite function $\rho(\boldsymbol{x})$ defined on a compact set $\mathbb{D}$ containing the origin, there exists a class $\mathcal{K}$ function $\Upsilon()$ such that:

$$\rho(\boldsymbol{x}) \geq \Upsilon(|\boldsymbol{x}|), \qquad \forall \boldsymbol{x} \in \mathbb{D}$$

Definition 3.4 (Positive invariant set).

A set $\mathbb{A}$ is positive invariant for the discrete nonlinear system $\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k))$ if:

$$\boldsymbol{x}(k) \in \mathbb{A}\ and\ \boldsymbol{x}(k+1) \in \mathbb{A}$$

Definition 3.5 (Asymptotic stability).

The steady state $\boldsymbol{xs}$ of a nonlinear discrete system $\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k))$ is asymptotically stable on $\mathbb{X}$, where $\mathbb{X}$ has $\boldsymbol{xs}$ in its interior, if there exist a $\Upsilon()$ such that for any $\boldsymbol{x} \in \mathbb{X}$, all solutions $\Phi(k; \boldsymbol{x})$ satisfy:

$$\Phi(k; \boldsymbol{x}) \in \mathbb{X}, |\Phi(k; \boldsymbol{x}) - \boldsymbol{xs}| \leq \Upsilon(|\boldsymbol{x} - \boldsymbol{xs}|, k) \qquad \forall\ k \in \mathbb{I}_{>0}$$

Where $\mathbb{I}_{>0}$ represents positive integers.

Definition 3.6 (Lyapunov function)

A function $V: \mathbb{R} \longrightarrow \mathbb{R}_{\geq 0}$ is said to be Lyapunov function for the nonlinear discrete system in the set $\mathbb{X}$ if there exists $\Upsilon i()$, where $i \in \{1,2,3\}$ such that for any $\boldsymbol{x} \in \mathbb{X}$

$$\Upsilon 1(|\boldsymbol{x}|) \leq V(\boldsymbol{x}) \leq \Upsilon 2(|\boldsymbol{x}|); \qquad V\big(\boldsymbol{x}(k+1)\big) - V\big(\boldsymbol{x}(k)\big) \leq -\Upsilon 3(|\boldsymbol{x}|)$$

Lemma 3.2(Lyapunov function and asymptotic stability). Consider a set $\mathbb{X}$ that is positive invariant for the nonlinear discrete system $\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k))$. The steady state $\boldsymbol{x}s$ is an asymptotically stable equilibrium point for the system if and only if there exists a Lyapunov function V on $\mathbb{X}$ such that V satisfies the properties described above (Definition 3.6).

Assumptions.

I. The stage cost[1] and nonlinear model are continuous.

II. There is weak controllability. Therefore, there exists $\Upsilon(): \mathbb{R} \longrightarrow \mathbb{R}_{\geq 0}$ so that for each $\boldsymbol{x} \in \mathbb{X}$ there exists a feasible $\boldsymbol{u}$ with:

$$|\boldsymbol{u} - \boldsymbol{u}s| \leq \Upsilon(|\boldsymbol{x} - \boldsymbol{x}s|)$$

Let define deviation variables as follows:

$$\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{x}s \qquad (3.1)$$
$$\boldsymbol{u}' = \boldsymbol{u} - \boldsymbol{u}s \qquad (3.2)$$

Where $\boldsymbol{x}', \boldsymbol{x}\ and\ \boldsymbol{x}s$ represent the deviation state vector, the absolute state vector and a steady state vector respectively. For the case of the input ($\boldsymbol{u}$) the definitions are the same (e.g. $\boldsymbol{u}'$ is the deviation input vector).

The deviation input vector will be calculated by a state feedback control law:

$$\boldsymbol{u}' = \boldsymbol{K}\boldsymbol{x} \qquad (3.3)$$

Where $\boldsymbol{K}$ is a controller matrix.

[1] Defined as the sum of the objective function for the entire prediction horizon. When the objective function is related to the economics of the processes, the stage cost is known as an Economic Stage Cost.

The nonlinear dynamic discrete system is defined as:

$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k)) \qquad (3.4)$$

Where $\boldsymbol{\theta}(k)$ represents a vector of parameters and the equations 3.1 and 3.2 can be applied to transform it into deviation form. Indeed, equations 3.1 and 3.2 can be applied also to the constraints and stage cost to transform them into deviations' variables form. In this system, $\boldsymbol{x}(k+1)$ represents the state vector at the next step; the sampling time T is used as a discretization time. In addition, $\boldsymbol{x} \epsilon \mathbb{X} \subseteq \mathbb{R}^n$ and $\boldsymbol{x}s \epsilon \mathbb{X}s \subset \mathbb{X}$. For the manipulated variable, $\boldsymbol{u} \epsilon \mathbb{U} \subset \mathbb{R}^m$ and $\boldsymbol{u}s \epsilon \mathbb{U}s \subset \mathbb{U}$. For the parameters $\boldsymbol{\theta} \epsilon \mathbb{Q} \subset \mathbb{R}^q$

The constraints are defined as:

$$E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k), \boldsymbol{P}(k)) \epsilon \mathbb{E} \qquad k \epsilon \mathbb{I}_{>0} \qquad (3.5)$$

Where the compact set $\mathbb{E}$ is defined as $\mathbb{E} \subseteq \mathbb{X}$ x $\mathbb{U}$ x $\mathbb{P}$ x $\mathbb{Q}$ and represents the admissible set for the constraints. Moreover, the set $\mathbb{Z}$ is defined as $\mathbb{Z} \subseteq \mathbb{X}$ x $\mathbb{U}$x $\mathbb{Q}$. The constraints' set $Z(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k))$ represents all common constraints that are enforced in standard MPC formulations such as lower and upper bounds in the states and inputs including average constraints in these variables. In addition, Z may also include upper and lower bounds in the parameters. The matrix ***P(k)*** is a positive define matrix used to define a Lyapunov function for the problem. Detail explanations on the calculation of ***P*** are provided in the next subsection.

$\mathbb{Z}$ does not include the constraints related to ***P*** , namely, $(Z(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k)) \epsilon \mathbb{Z} \quad k \epsilon \mathbb{I}_{>0})$, hence, $\mathbb{Z} \subset \mathbb{E}$.

The economic stage cost is defined as:

$$L = l(\boldsymbol{x}(k), \boldsymbol{u}(k)) \qquad (3.6)$$

Where $l(\boldsymbol{x}(k), \boldsymbol{u}(k)): \mathbb{Z} \longrightarrow \mathbb{R}$

Finally, the state transition map for the nonlinear discrete system is:

$$f(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k)): \mathbb{Z} \longrightarrow \mathbb{R}^n$$

The best steady state solution for a fixed parameter vector comes from:

$$\min_{\boldsymbol{u}} l(\boldsymbol{x}, \boldsymbol{u})$$

s.t.

$$0 = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta_{fix}}) \qquad (3.7\text{ a})$$
$$Z(\boldsymbol{x}, \boldsymbol{u})$$

Where $\boldsymbol{u}s$ and $\boldsymbol{x}s$ represents the best steady state pair and $0 = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}_{fix})$ is the steady state model. Alternatively, the steady state problem may be stated as:

$$\min_{\boldsymbol{u}} l(\boldsymbol{x}s, \boldsymbol{u}s)$$

s.t.

$$0 = f(\boldsymbol{x}s, \boldsymbol{u}s, \boldsymbol{\theta}_{fix}) \qquad (3.7\text{ b})$$
$$Z(\boldsymbol{x}s, \boldsymbol{u}s)$$

Where $\boldsymbol{u} = \boldsymbol{u}s \; and \; \boldsymbol{x} = \boldsymbol{x}s$. It is assumed that there is unique solution.

Our proposed dynamic optimization problem is defined as:

$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P},\boldsymbol{\theta}} \sum_{k=1}^{N} l(\boldsymbol{x}(k),\boldsymbol{u}(k)) \tag{3.8}$$

s.t.

$$\boldsymbol{x}(0) = \boldsymbol{x}o$$
$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k),\boldsymbol{u}(k),\boldsymbol{\theta}(k))$$
$$E(\boldsymbol{x}(k),\boldsymbol{u}(k),\boldsymbol{P}(k),\boldsymbol{\theta}(k))$$
$$0 = f(\boldsymbol{x}s(k),\boldsymbol{u}s(k),\boldsymbol{\theta}(k))$$

Where N and $\boldsymbol{x}o$ represent the finite horizon and initial state respectively. Moreover, the steady state and parameters are fixed for all the horizon length N, implying that $\boldsymbol{u}s(1) = \boldsymbol{u}s(2) = \cdots \boldsymbol{u}s(N)$ and $\boldsymbol{\theta}(1) = \boldsymbol{\theta}(2) = \cdots \boldsymbol{\theta}(N)$.

Since the problem is formulated in deviation variables a steady state model has to be used to calculate the reference for the deviations. However, it is important to notice that the steady state may change at each execution of the EMPC controller since **u**s is a decision variable. This is a key novelty in our proposed algorithm since previously proposed approaches have used a time invariant set point obtained off-line. This dynamic optimization problem is solved in a receding horizon fashion, implying that the first element of the solution vector $\boldsymbol{u}(k) = \boldsymbol{u}'(k) + \boldsymbol{u}s(k) = [\boldsymbol{u}(1,k),\boldsymbol{u}(2,k),\boldsymbol{u}(3,k) \ldots \boldsymbol{u}(N,k)]$, namely $\boldsymbol{u}(1,k)$, and the computed $\boldsymbol{\theta}(1,k)$ are applied to the dynamic discrete system. Then, after applying this control action, the vector of states $\boldsymbol{x}(k+1,k)$ is used at the next step for solving the problem again where the first element of the vector $\boldsymbol{x}(k+1,k)$ is measured.

The feedback control law is defined as: $\boldsymbol{u}(k)^* = \boldsymbol{u}(1,k)$

In the next subsection, special constraints are introduced to ensure stability of the closed loop algorithm.

For the case where the parameters are constant, the dynamic optimization problem can be reduced to:

$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P}} \sum_{k=1}^{N} l(\boldsymbol{x}(k), \boldsymbol{u}(k)) \tag{3.9}$$

s.t.

$$\boldsymbol{x}(0) = \boldsymbol{x}o$$
$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k))$$
$$E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{P}(k))$$
$$0 = f(\boldsymbol{x}s(k), \boldsymbol{u}s(k))$$

Where once again, the problem is solved in a receding horizon manner.

3.2.2 Constraints to ensure asymptotic stability

The stability in our formulation comes from the implementation of 4 "special" constraints within the optimization problem presented in the previous section. In this section we are discussing each one of these constraints and their effect in our formulation.

The constraints' function $Z(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k))$ used in the previous subsection represents all common constraints that are enforced in standard MPC formulations such as lower and upper bounds in the states and inputs including average constraints in these variables. In addition, Z may also include upper and lower bounds in the parameters.

On the other hand the complete set of constraints, $E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k), \boldsymbol{P}(k))$ encompasses not only $(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k))$, but also four additional constraints to ensure stability as follows.

I. Positive definite **P**

This constraint enforces that $\boldsymbol{P}(k)$ that is a decision variable of the optimization problem formulated in equation (3.8) must be positive definite at each step. Namely, it has to follow the properties of Definition 3.1 ($\boldsymbol{P}$(k) must be equal to its transpose and bigger than cero).

II. "Set Point Constraint"

The set point constraint is defined as follows:

$$\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big)^T\boldsymbol{P}(k-1)\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big) < \alpha\big(\boldsymbol{x}(k-1)-\boldsymbol{x}s(k-1)\big)^T\boldsymbol{P}(k-1)\big(\boldsymbol{x}(k-1)-\boldsymbol{x}s(k-1)\big) \quad (3.10)$$

The set point constraint has a quadratic form and it tests the negativity of the rate of change of the Lyapunov function with respect to changes in set point. As it can be seen, the LHS and RHS share the same value of **P**, as a result, the only way to satisfy this constraint is to find a new steady state vector $\boldsymbol{x}s(k)$ and/or a state trajectory $\boldsymbol{x}(k)$ that decreases the LHS. Furthermore, this constraint has a scalar factor (α) in the RHS which accounts for the adaptation rate that is going to be explained later. When no adaptation is needed, the value of α is one.

As stated in the previous section, the set point constraint plays an important role in the performance of the proposed algorithm. Since $\boldsymbol{u}s(k)$ and $\boldsymbol{\theta}(k)$ are decision variables, the set point $\boldsymbol{x}s(k)$ is determined by these two variables from the steady state model given by equation (3.8).

Additionally, this constraint is essential for determining the rate of adaptation of parameters as explained later in this thesis. If there is a mismatch between the model and plant parameters, the optimization will force $\boldsymbol{\theta}(k)$ to change so as to satisfy the set point constraint through a new value of $\boldsymbol{x}s(k)$.

III. "P Constraint"

The P constraint is defined as:

$$\big(\boldsymbol{x}(k) - \boldsymbol{x}s(k)\big)^T \boldsymbol{P}(k)\big(\boldsymbol{x}(k) - \boldsymbol{x}s(k)\big) \leq \big(\boldsymbol{x}(k) - \boldsymbol{x}s(k)\big)^T \boldsymbol{P}(k-1)\big(\boldsymbol{x}(k) - \boldsymbol{x}s(k)\big) \tag{3.11}$$

To reduce potential conservatism, the matrix **P** defining the Lyapunov function of the problem is a decision variable of the problem and thus it is time varying. This constraint is also of quadratic form and it tests set invariance with respect to changes in the value of $\boldsymbol{P}$.The $\boldsymbol{P}$ at the LSH is at the current time whereas the $\boldsymbol{P}$ at the RHS is at the previous time. This implies that:

$$\boldsymbol{P}(k) < \boldsymbol{P}(k-1) \tag{3.12}$$

The P constraint limits the changes of the optimization variable $\boldsymbol{P}$ by forcing all the $\boldsymbol{P}(i),\ \ where\ i > 1$, to be contained in an initial region ($\boldsymbol{P}(1)$).

This condition ensures that the deviations will remain within the initial invariant set which is defined as:

$$\big(\boldsymbol{x}(0) - \boldsymbol{x}s(1)\big)^T \boldsymbol{P}(0)\big(\boldsymbol{x}(0) - \boldsymbol{x}s(1)\big) \tag{3.13}$$

Where $\boldsymbol{x}(0)$ represents the initial measurement, **P**(0) represents an initial value of the P matrix and $\boldsymbol{x}s(1)$ the computed steady state value at the first EMPC iteration.

IV. "Lyapunov Based Robust Stability Constraint"

This constraint explicitly addresses not only the asymptotic stability of the system but the robustness.

Linearizing the nonlinear system around some operation point and transforming it into deviation variables gives:

$$\boldsymbol{x}'(k+1) = \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{x}'(k) + \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}'(k) \qquad (3.14)$$

Using 3.3 we obtain:

$$\boldsymbol{x}'(k+1) = \boldsymbol{A_{CL}}(\boldsymbol{\theta}, k)\boldsymbol{x}'(k) \qquad (3.15)$$

Where $\boldsymbol{A_{CL}}(\boldsymbol{\theta}, k) = \boldsymbol{A}(\boldsymbol{\theta}) + \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{K}(k)$

The plant model is represented by a convex set $\mathbb{A}_{\mathbb{T}}$ of linear plants with M vertices:

$$\mathbb{A}_{\mathbb{T}} = \{\boldsymbol{A_{CL1}}(\boldsymbol{\theta}, k), \boldsymbol{A_{CL2}}(\boldsymbol{\theta}, k) \dots \boldsymbol{A_{CLM}}(\boldsymbol{\theta}, k)\} = \sum_{i=1}^{M} \zeta_i [A_{\boldsymbol{CLi}}(\boldsymbol{\theta}, k)]; \quad \forall \zeta_i \geq 0, \sum_{i=1}^{M} \zeta_i = 1 \qquad (3.16)$$

Thus the representation 3.16, describes the plant model by a polytope of matrices. Each element of the polytope, $\boldsymbol{A_{CLi}}(\boldsymbol{\theta})$, can be derived from the definition in 3.15. Using 3.16 it follows that the plant model can be approximated as follows:

$$\boldsymbol{x}'(k+1) = \sum_{i=1}^{M} \zeta_i [A_{\boldsymbol{CLi}}(\boldsymbol{\theta}, k)]\, \boldsymbol{x}'(k) \qquad (3.17)$$

Once the plant model has been formulated, a candidate Lyapunov function is:

$$V(k) = \boldsymbol{x}'(k)^T \boldsymbol{P}(k)\boldsymbol{x}'(k) \qquad (3.18)$$

This quadratic Lyapunov function has all the properties stated in the Definition3.6. Then, for 3.18 to be a Lyapunov function of the problem the following constraint has to be imposed:

$$V(k+1) - V(k) < 0 \qquad (3.19)$$

To ensure this constraint for the worst case with the polytopic model defined in 3.16, the first term of the LHS is derived from a maximization problem as follows:

$$V(k+1) = \underset{\zeta}{\text{argmax}}[\boldsymbol{x}'(k+1)^T \boldsymbol{P}(k)\boldsymbol{x}'(k+1)]$$

$$\begin{aligned} &\text{s.t.} \\ &\boldsymbol{x}'(k+1) = \textstyle\sum_{i=1}^{M} \zeta_i [A_{\boldsymbol{CL}i}(\boldsymbol{\theta}, k)]\, \boldsymbol{x}'(k) \\ &\textstyle\sum_{i=1}^{M} \zeta_i = 1 \\ &0 \le \zeta_i \le 1 \end{aligned} \qquad (3.20)$$

The use of the maximal value of V(k+1) guaranties robustness with respect to the polytopic model since it is the worst case scenario for inequality 3.19.

The four constraints listed in this subsection are added to the set of constraints given by Z to form the total set of constraints E that is ultimately used in the proposed EMPC algorithm. A key difference of the proposed algorithm is that stability is ensured on-line rather than in an off-line fashion as done in previous approaches by using Dissipativity or Lyapunov arguments. A main advantage of this on-line approach is that it does not require restrictive terminal conditions and the use of fixed steady state solutions as required in the off-line stability proofs previously reported. On the other hand the on-line computation of the stability related constraints presented in this chapter results in significant increase in computational effort.

3.3 APPLICATION TO REACTOR SYSTEM

The proposed EMPC is applied to a reactor system example presented by Rawlings et al (2012). The dimensionless dynamic equations, process constraints, values of parameters, best steady state, and the designation of the economic stage cost were taken from Rawling's work.

The reactor system consists of an isothermal CSTR with parallel reactions.

$$P_0 + B \longrightarrow P_1$$
$$P_1 + B \longrightarrow P_2$$

The dimensionless nonlinear mass balance equations are described by:

$$\frac{dx_1}{dt} = u_1 - x_1 - \sigma_1 x_1 x_2 \qquad (3.21)$$
$$\frac{dx_2}{dt} = u_2 - x_2 - \sigma_1 x_1 x_2 - \sigma_2 x_2 x_3 \qquad (3.22)$$
$$\frac{dx_3}{dt} = -x_3 + \sigma_1 x_1 x_2 - \sigma_2 x_2 x_3 \qquad (3.23)$$
$$\frac{dx_4}{dt} = -x_4 + \sigma_2 x_2 x_3 \qquad (3.24)$$

Where x_1, x_2, x_3, x_4 represent the concentrations of P_0,B, P_1, P_2 respectively and u_1, u_2 represent the flow rates of P_0 and B. For this example the parameters σ_1 and σ_2 are constant and have a value of 1 and 0.4 respectively. The inputs u_i have upper and lower bounds.

$$0 \leq u_1 \leq 1 \qquad (3.25)$$
$$0 \leq u_2 \leq 10 \qquad (3.26)$$

The best steady state for these bounds is:

$\boldsymbol{xs} = [0.3874, 1.5811, 0.3752, 0.2373]^T$ and $\boldsymbol{us} = [1, 2.4310]^T$

We assume the same sampling time as described in Rawlings et al (2012) of T=0.1.

For this particular formulation, the economic stage cost is obtained from the assumption that profit is directly related to the concentration of P_1which is the most valuable product. Thus, the cost to be maximized is:

$$l = -x_3$$

The dynamic optimization problem can be stated as follows:

$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P}} \sum_{k=1}^{N} -x_3 \tag{3.26}$$

s.t.

$$\boldsymbol{x}(0) = \boldsymbol{x}o$$
$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k))$$
$$E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{P}(k))$$
$$0 = f(\boldsymbol{x}s(k), \boldsymbol{u}s(k))$$

Where, as it was previously discussed, $E()$ encompasses all the constraints including the four stability related constraints presented in the previous subsection. It is also important to note that the adaptation problem is not considered and it will be discussed later in Chapter 4.

Since the on-line calculation of the stability related constraints resulted in long computations and since the optimization was executed with *fmincon* in the Matlab environment, we had to use relatively small values of the prediction horizon *N*. It is expected that the use of faster hardware and software should permit the use of longer horizon.

One of the disadvantages found with the choice of small N is that the optimizer produced transient values of economic costs that were significantly different than the final best steady state cost. As a result of that, it was found that the program was not converging to the best steady state cost. To compensate for this situation, it was decided to add to the original dynamic cost a steady state economic cost with a weight.

Considering the definitions of deviation variables, the stage cost is modified as follows:

$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P}} \sum_{k=1}^{N} l\big(\boldsymbol{x}'(k), \boldsymbol{u}'(k)\big) + Ql(\boldsymbol{x}s(k), \boldsymbol{u}s(k))$$

Now, it should be noticed that the dynamic economic cost term is written as a function of the deviation variables and a static one that is function of the steady state. Q is a tunning parameter which has values of $Q \geq 1$. Note that for linear cost functions, as in this case, when $Q = 1$, the stage cost reduces to the stage cost in 3.8. Furthermore, if the steady state profile is constant the stage cost becomes purely economic because the second term in the cost is constant.

One possibility for systematically choosing the weight Q is to relate it to the frequency of the disturbances. For example, if disturbances are of slow frequency Q will be chosen correspondingly large so as to drive the system to the best steady state. For slow disturbances, it is convenient to converge to the best steady sate since the system most of the time will be there. For disturbances with high frequency, a smaller value of Q will be chosen so as to keep the system in a dynamic mode, taking advantage of the transient of the process. Although a systematic approach for choosing Q is beyond the scope of the current proposal, some simulations are presented below for different values of Q to illustrate its effect on the closed loop performance.

For the reactor example, the EMPC becomes:

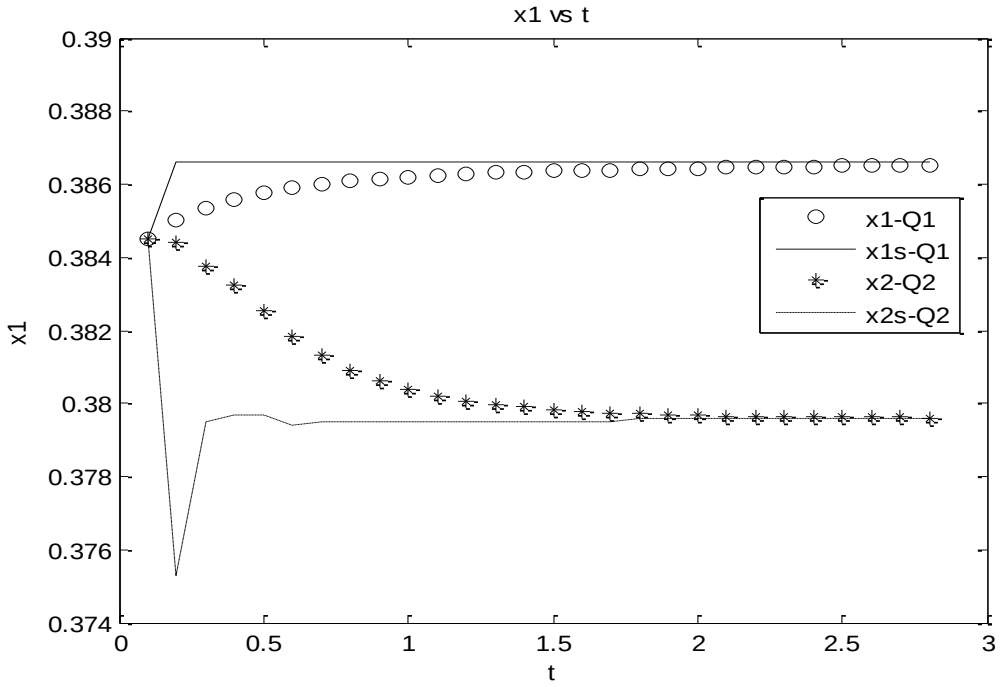
$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P}} \sum_{k=1}^{N} -(x'_3 + Qxs_3) \tag{3.28}$$

s.t.

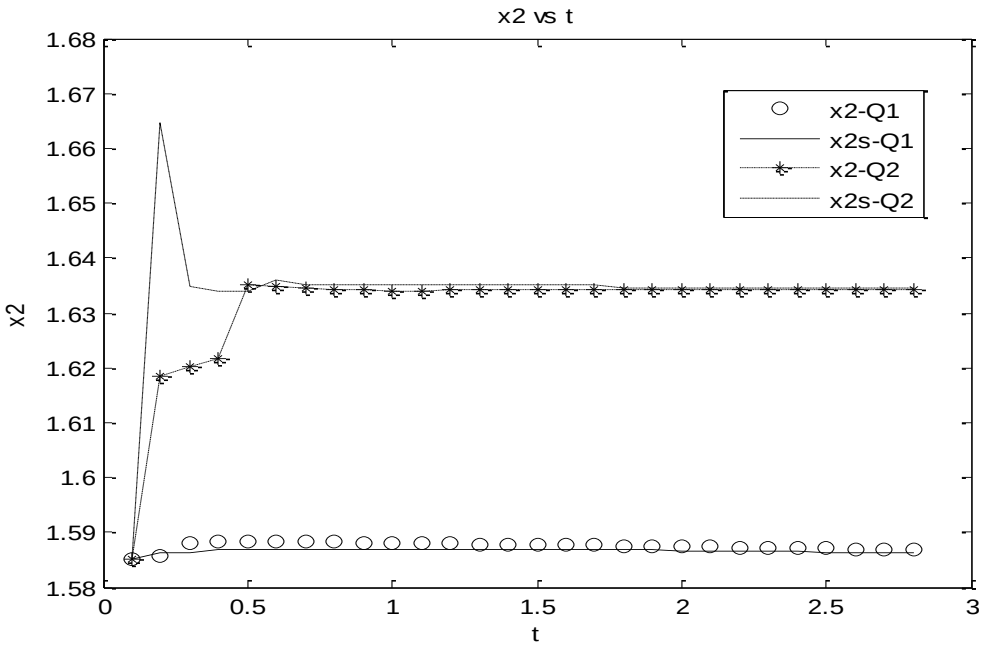
$$\boldsymbol{x}(0) = \boldsymbol{x}o$$
$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k))$$
$$E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{P}(k))$$
$$0 = f(\boldsymbol{x}s(k), \boldsymbol{u}s(k))$$

We analyzed the EMPC formulation for two cases with different values of Q. The horizon had a value of 15 and it was the same in both cases. Additionally, the problem was solved using fmincon for optimization and ode45 and fsolve for the differential equations and steady state model respectively.

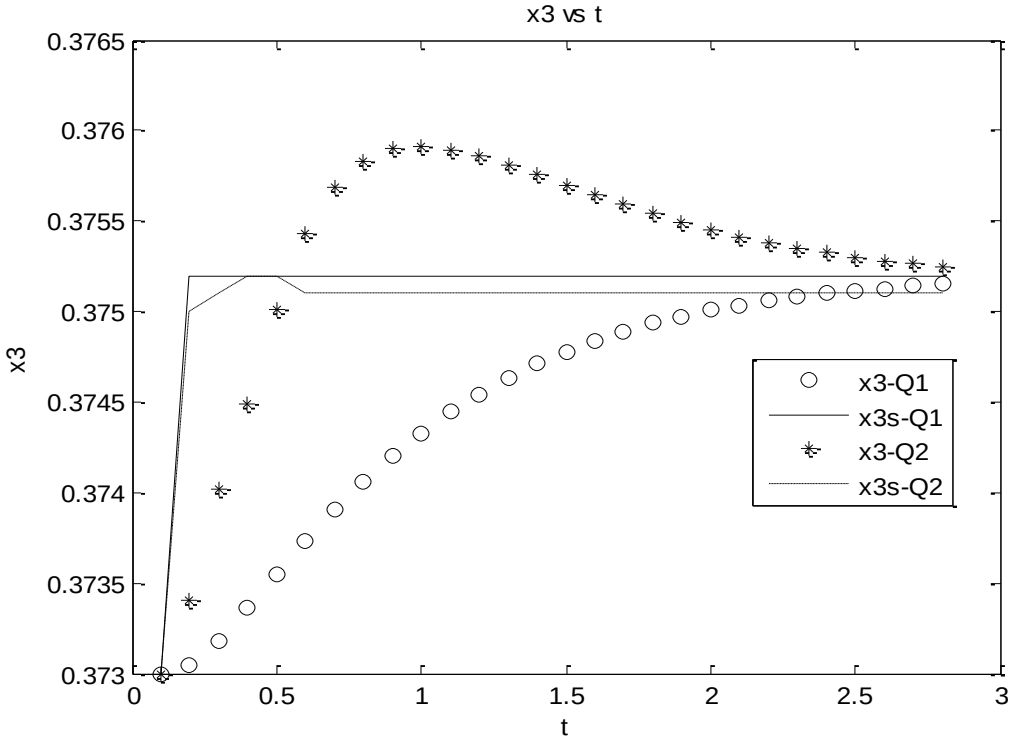
In Figure 3.1, the closed loop state trajectories are shown for some initial condition (suboptimal operation at steady state) and different values of Q, Q_1=45 and Q_2=6. A non-zero initial condition can be viewed as equivalent to a rejection disturbance problem where the disturbance causes an initial deviation from steady state conditions and the system is then returned to steady state by the controller.



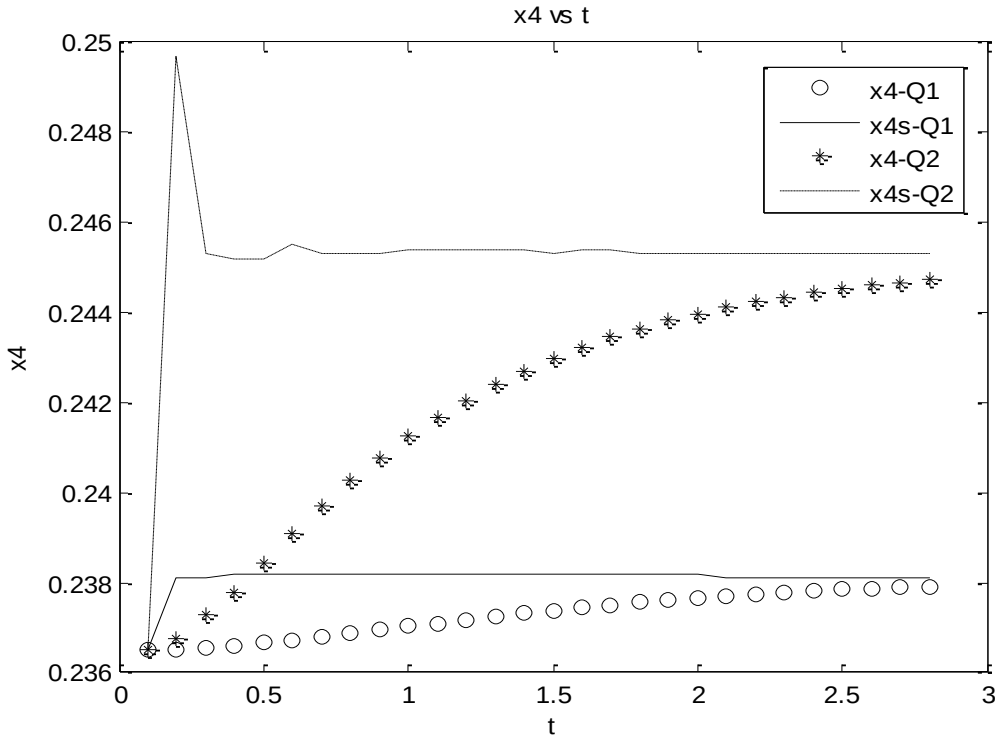
(a)


x2 vs t
x2
t
x2-Q1
x2s-Q1
x2-Q2
x2s-Q2
1.68
1.67
1.66
1.65
1.64
1.63
1.62
1.61
1.6
1.59
1.58
0
0.5
1
1.5
2
2.5
3

(b)


x3 vs t
x3
t
x3-Q1
x3s-Q1
x3-Q2
x3s-Q2
0.3765
0.376
0.3755
0.375
0.3745
0.374
0.3735
0.373
0
0.5
1
1.5
2
2.5
3

(c)



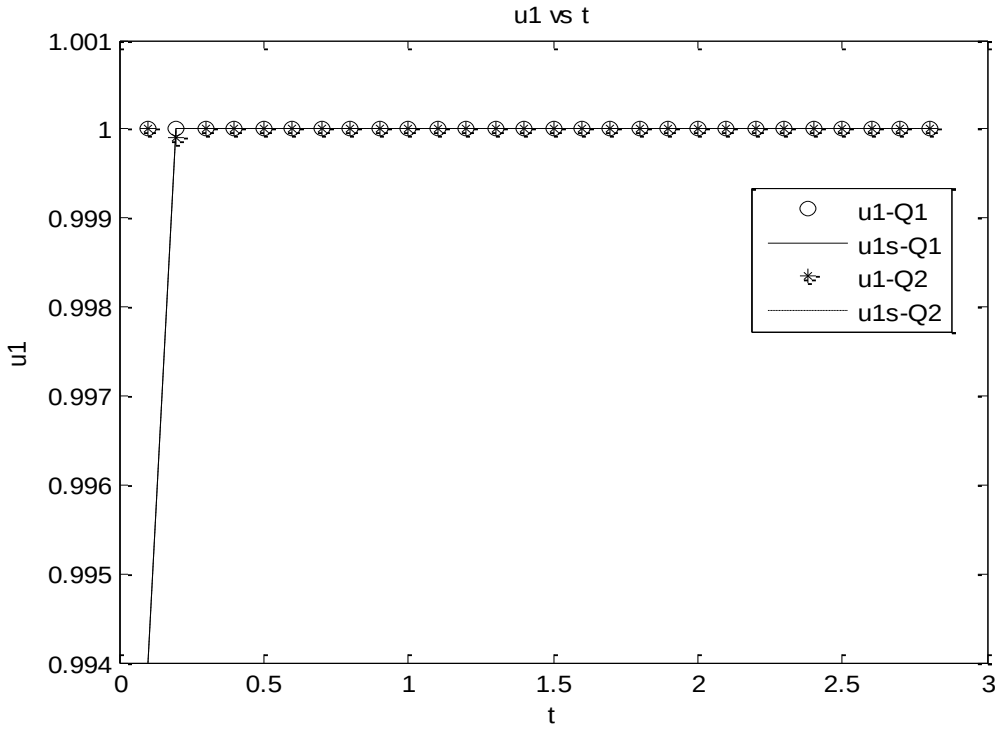
(d)

Fig. 3.1: Closed loop state (a), (b), (c) and (d) profiles for isothermal reactor.

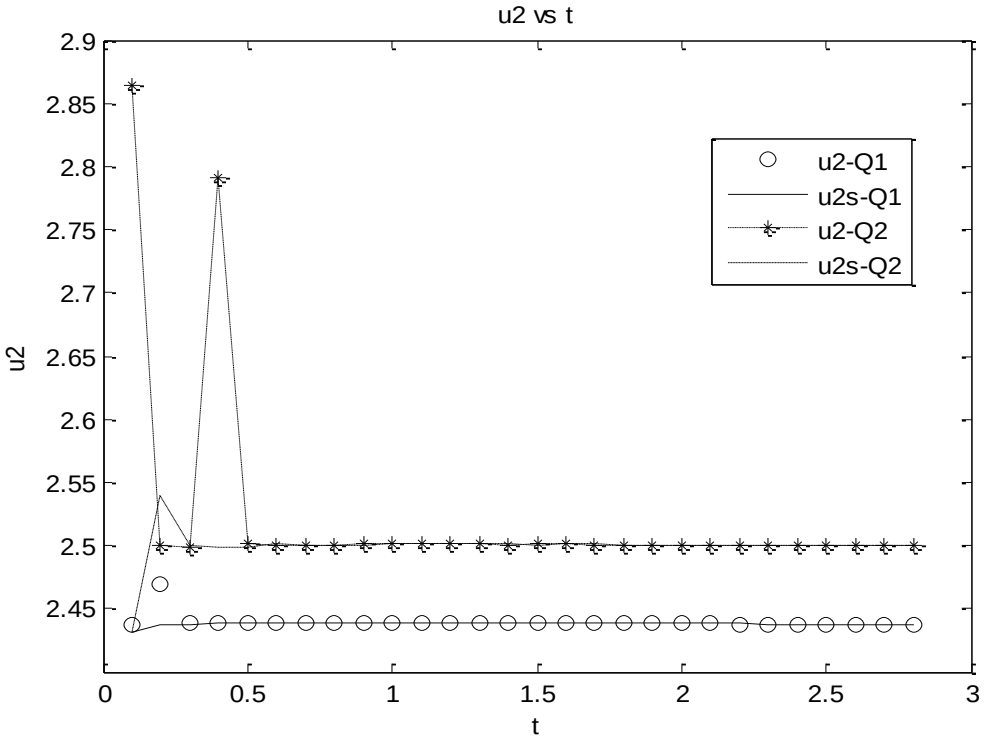
From Figure 3.1, it is evident that the system is closed loop stable for both values of Q. For Q_2 the steady state changes at the beginning of the simulation and settle down around t = 0.7, whereas for Q_1 the computed steady state is practically the same during the simulation. It is also clear that two different steady states profiles are generated for the different values of Q.

From the graphic of x_3 in Figure 3.1, it is evident that Q_2 results in a better stage cost. This fact reinforces the idea that the choice of Q has a very significant effect on performance. However, regardless of the value of Q , the value of x_3 converged to the best economic steady state which is equal to x_3=0.3752.

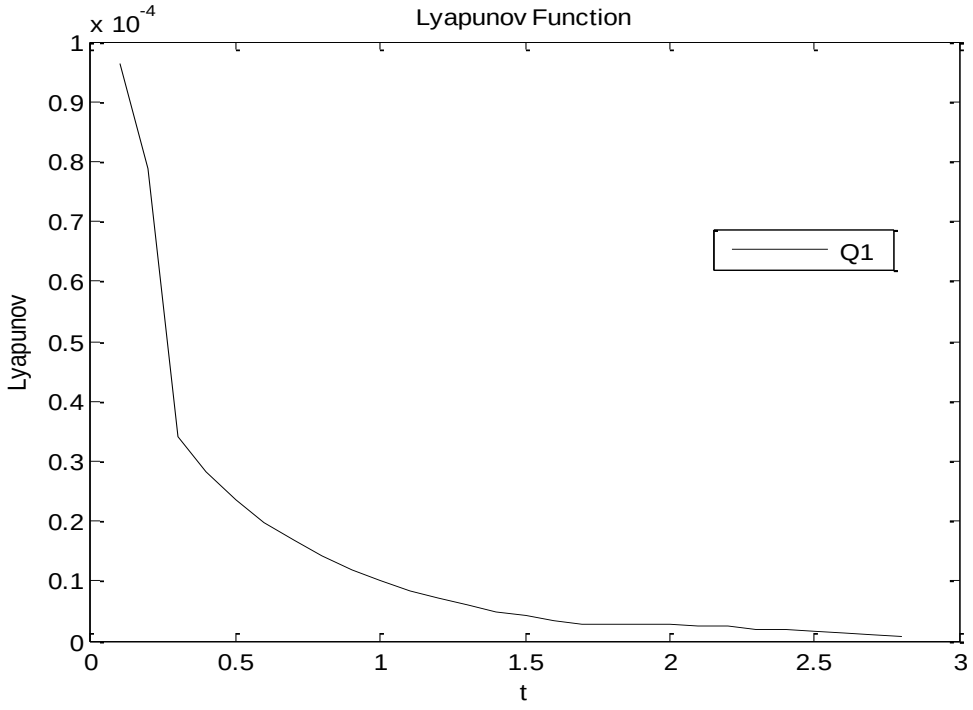
Figure 3.2 shows the closed loop input and quadratic Lyapunov function ($\boldsymbol{x}'(k)^T \boldsymbol{P}(k)\boldsymbol{x}'(k)$) of the plant.


u1 vs t
u1
t
u1-Q1
u1s-Q1
u1-Q2
u1s-Q2
1.001
1
0.999
0.998
0.997
0.996
0.995
0.994
0
0.5
1
1.5
2
2.5
3

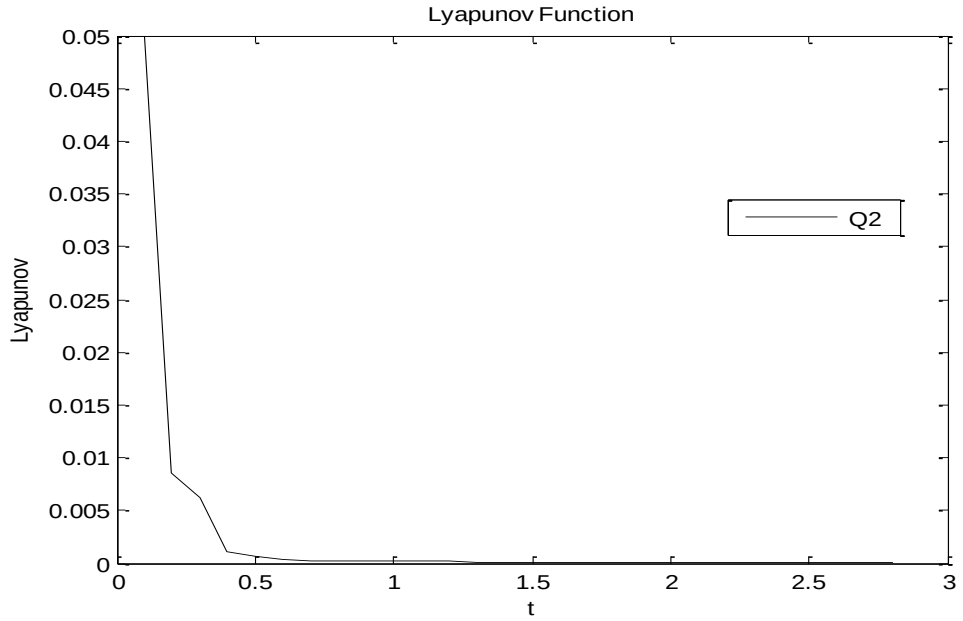
(a)


u2 vs t
u2
t
u2-Q1
u2s-Q1
u2-Q2
u2s-Q2
2.9
2.85
2.8
2.75
2.7
2.65
2.6
2.55
2.5
2.45
0
0.5
1
1.5
2
2.5
3

(b)



(c)



(d)

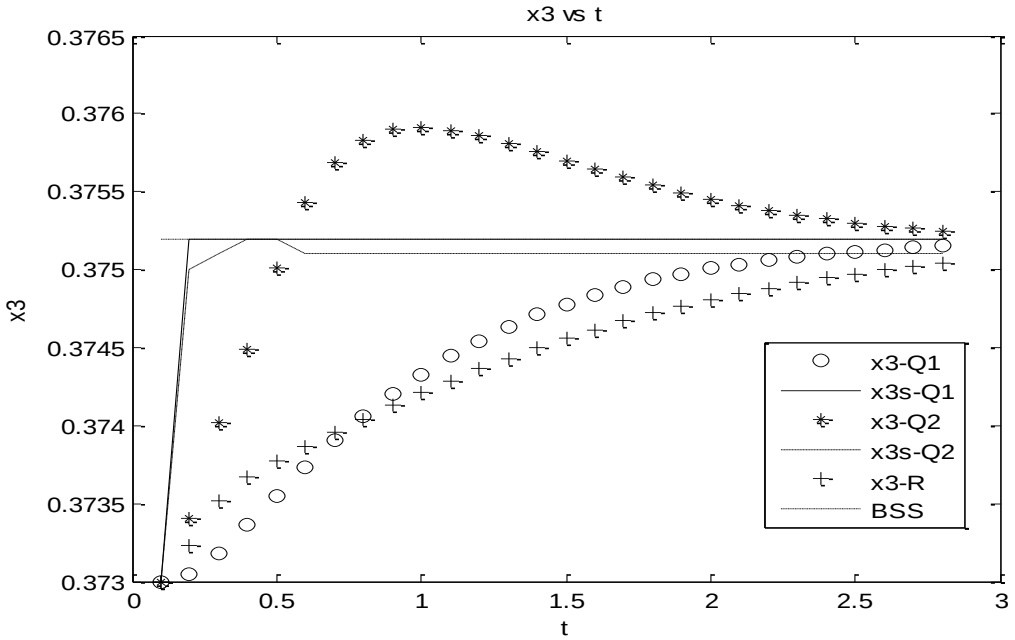
Fig. 3.2: Closed loop input (a) and (b),and Lyapunov (c) and (d) profiles for isothermal reactor

From Figure 3.2 (a) and (b) it is clear that the inputs complied with the imposed input constraints.

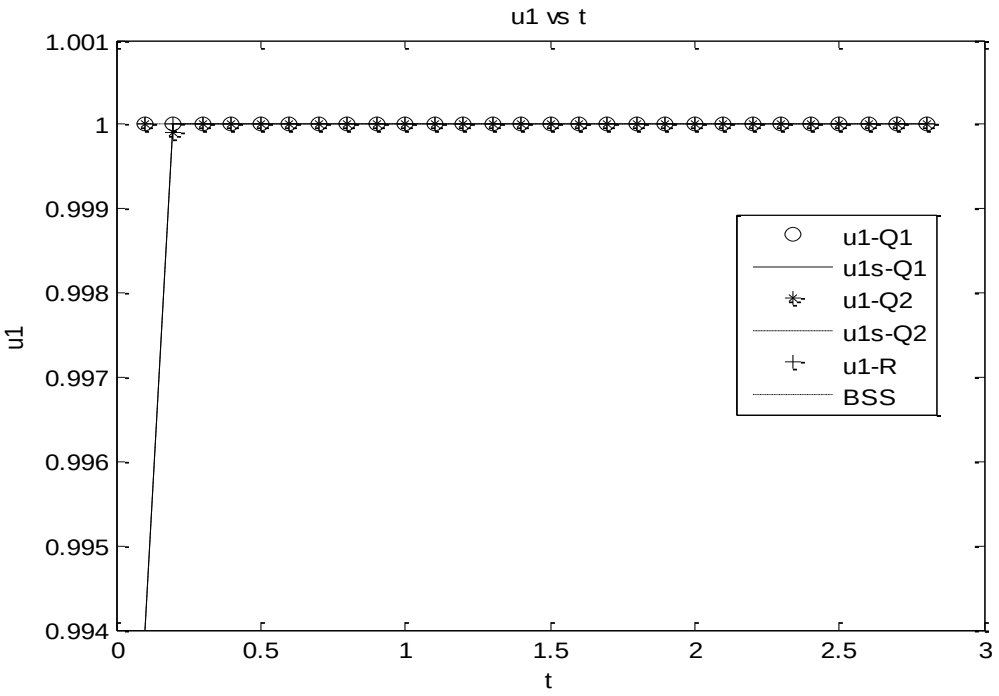
From 3.2 (c) and (d) it is also evident that for both Q values the polypote is a good approximation of the plant since the values of the Lyapunov functions calculated with the actual states obtained from the nonlinear simulated process model monotonically decreased with time. Moreover, the Lyapunov functions show that the non-zero initial condition is significant since their closed-loop behavior is strictly decreasing.

The Strongly Dual algorithm described in Rawlings et al (2012) was simulated and compared to our approach. The values for the tuning parameters, steady state and terminal constraints were taken from that paper. In addition, the proposed algorithm was solved using the same horizon length, optimizer and ODE solver's settings as in our case.

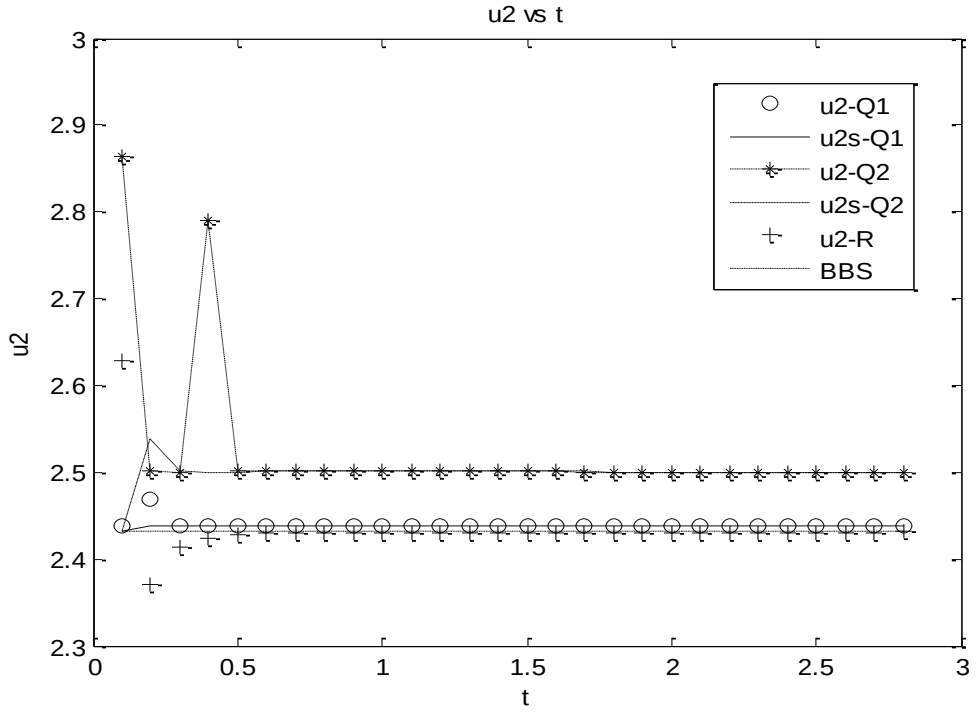
Fig. 3.3 shows the closed loop inputs and corresponding x_3 profiles. The profile for x_3 in the current approach with Q_1 is very similar to the one obtained by Rawlings, however, there is a clear difference when the current approach is executed with Q_2 .The flow P_0 (or u_1) is identical in all the simulations.



(a)



(b)



(c)

Fig. 3.3: Closed loop state (a) and (b), (c) inputs profiles with alternative formulation.

Table 3.1 shows the economic performance in terms of Average Cost (Closed-loop average of the objective function for the entire simulation). The profit obtained with our approach in these examples slightly outperform or are very similar to Rawlings' EMPC.

For Q_1, where the steady state has priority, our results are very similar to Rawlings. Nonetheless, for Q_2, where the transient is also important, our approach is better.

EMPC	Average Cost	% Improvement
Q_1	0.3745	0.0212
Q_2	0.3753	0.2412
Rawlings	0.3744	0

Table 3.1: Performance comparison between our formulation (Q1 and Q2) and the Rawlings formulation.

Moreover, the approach proposed in this thesis avoids some of the potentially limiting features of Rawlings' in that it does not require: (i) to perform off-line optimization to obtain the best steady state, (ii) to solve strong duality/dissipativity problems, and (iii) to use terminal constraints or convex quadratic terms to the cost function.

Chapter 4

Parameter Adaptation, Otto Reactor Simulation, Feasibility

4.1 INTRODUCTION

In this chapter we will illustrate the ability of the proposed algorithm to deal with time varying model parameters and to tackle more complex dynamic optimization problems. In section 4.2, the same example of a reactor with parallel reactions studied in section 3.3 is developed but in this case a parameter adaptation is implemented to account for changes in parameters. The solution of a Williams-Otto reactor which also has parallel reactions is reviewed in section 4.3. Finally, the numerical solution and the feasibility of these more complicated examples is discussed in section 4.4.

4.2 PARAMETER ADAPTATION

As previously discussed, the algorithm for parameter adaptation was shown by equation 3.8 and is as follows:

$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P},\boldsymbol{\theta}} \sum_{k=1}^{N} l(\boldsymbol{x}(k),\boldsymbol{u}(k)) \tag{4.1}$$

s.t.

$$\boldsymbol{x}(0) = \boldsymbol{x}o$$
$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k),\boldsymbol{u}(k),\boldsymbol{\theta}(k))$$
$$E(\boldsymbol{x}(k),\boldsymbol{u}(k),\boldsymbol{P}(k),\boldsymbol{\theta}(k))$$
$$0 = f(\boldsymbol{x}s(k),\boldsymbol{u}s(k),\boldsymbol{\theta}(k))$$

We applied this algorithm to the reactor problem in section 3.3 where one of the kinetic parameters (σ_1) is adapted. Constraints were identical to the previous study in terms of bounds in

inputs and horizon, except that in the current study σ_1 was assumed to be between bounds as follows:

$$1 \leq \sigma_1 \leq 1.03$$

The simulations were conducted with the tuning parameter Q_1. The resulting dynamic optimization problem is as follows:

$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P},\boldsymbol{\theta}} \sum_{k=1}^{N} -(x'_3 + Q_1 xs_3) \tag{4.2}$$

s.t.

$$\boldsymbol{x}(0) = \boldsymbol{x}o$$
$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\theta}(k))$$
$$E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{P}(k), \boldsymbol{\theta}(k))$$
$$0 = f(\boldsymbol{x}s(k), \boldsymbol{u}s(k), \boldsymbol{\theta}(k))$$

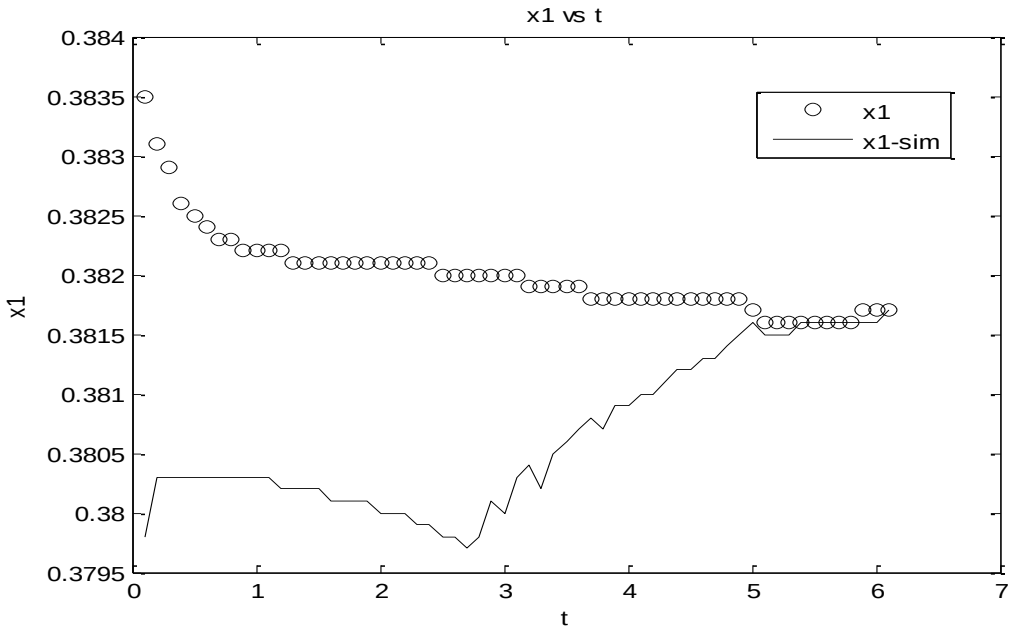
Where $E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{P}(k), \boldsymbol{\theta}(k))$ includes now the bounds for σ_1 in addition to all the constraints considered for the same case study in Chapter 3.

Considering equation 3.10:

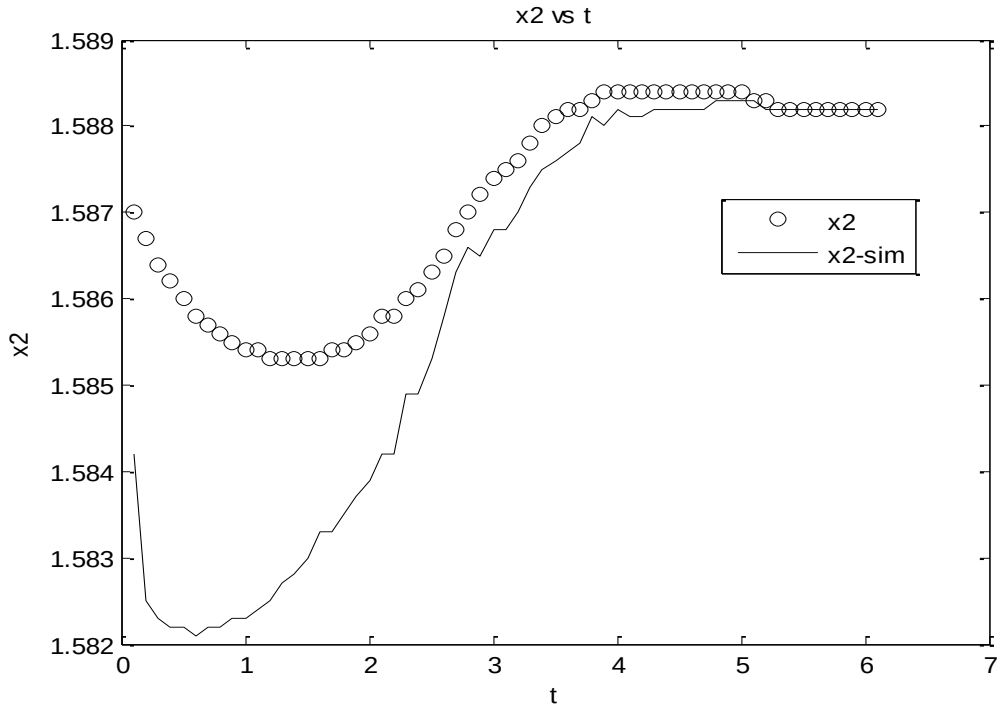
$$\big(\boldsymbol{x}(k) - \boldsymbol{x}s(k)\big)^T \boldsymbol{P}(k-1)\big(\boldsymbol{x}(k) - \boldsymbol{x}s(k)\big) < \alpha\big(\boldsymbol{x}(k-1) - \boldsymbol{x}s(k-1)\big)^T \boldsymbol{P}(k-1)\big(\boldsymbol{x}(k-1) - \boldsymbol{x}s(k-1)\big)$$

We can see that α plays an important role regarding the rate of adaptation because the contraction of the Lyapunov by manipulating **x**s(k) and consequently **θ**(k) must be greater for values of α smaller than unity. The smaller the value of α, the faster the adaptation rate. Nevertheless, if α is chosen so small, the problem may become infeasible. Therefore, α is a tuning parameter that must be selected carefully. The systematic selection of alfa is outside the scope of this work, but it can be tuned according to the feasibility of 3.10. For this particular problem α had a value of 0.9.

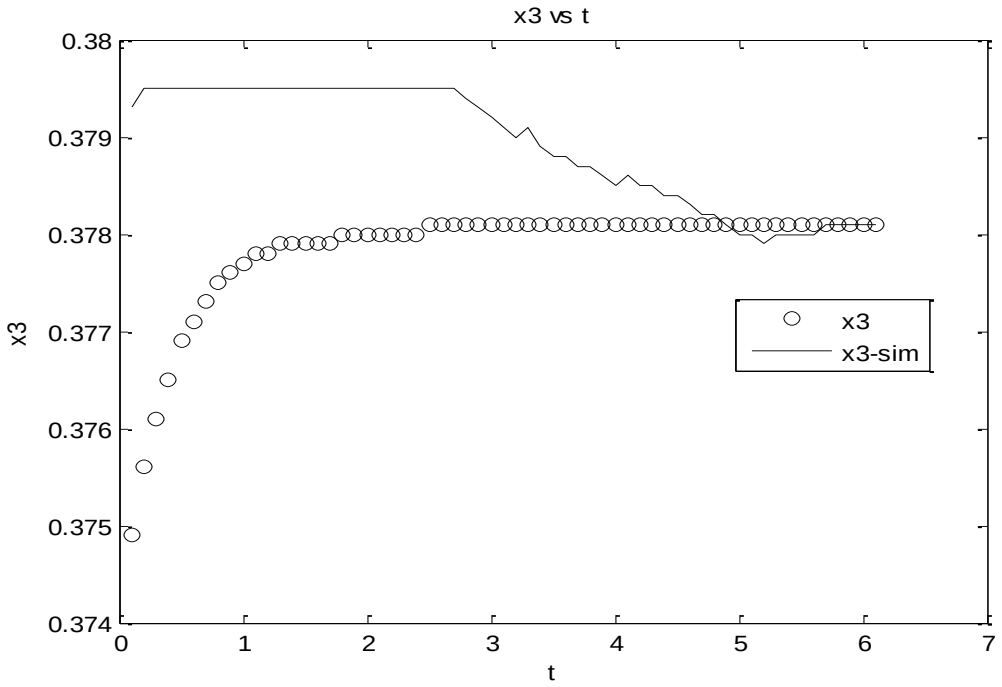
Figure 4.1 shows the closed loop state trajectories for a particular initial condition where an initial mismatch between the model and the plant in terms of the value of the parameterσ_1. The solid line represents the simulated values, whereas the circles represent the true measured values.



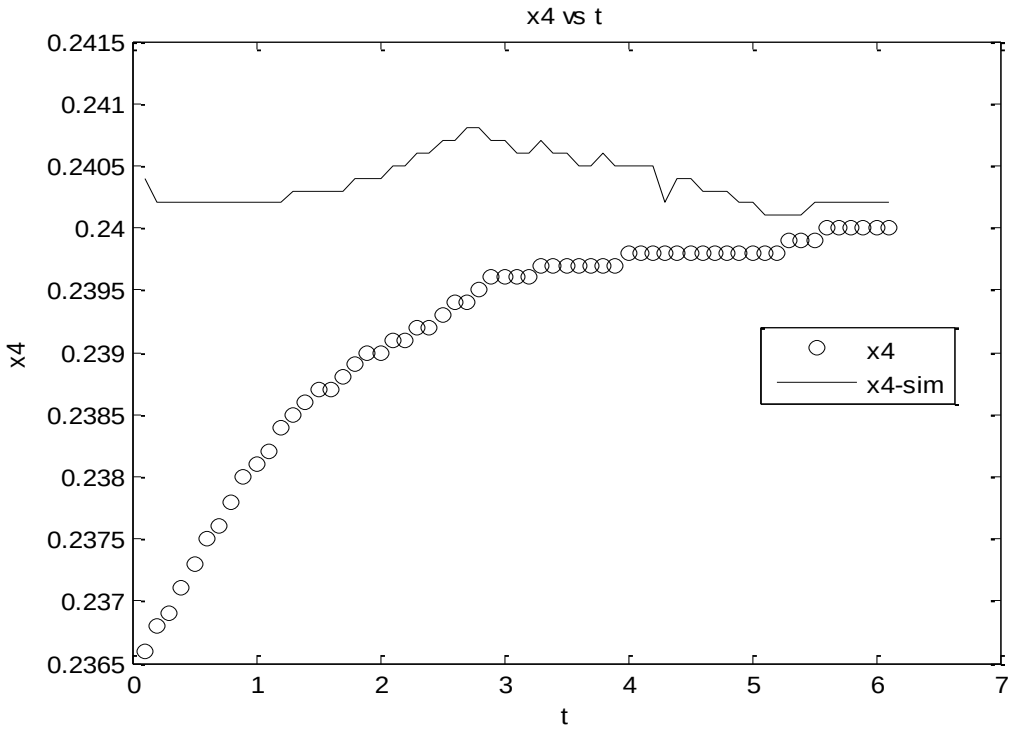
(a)



(b)



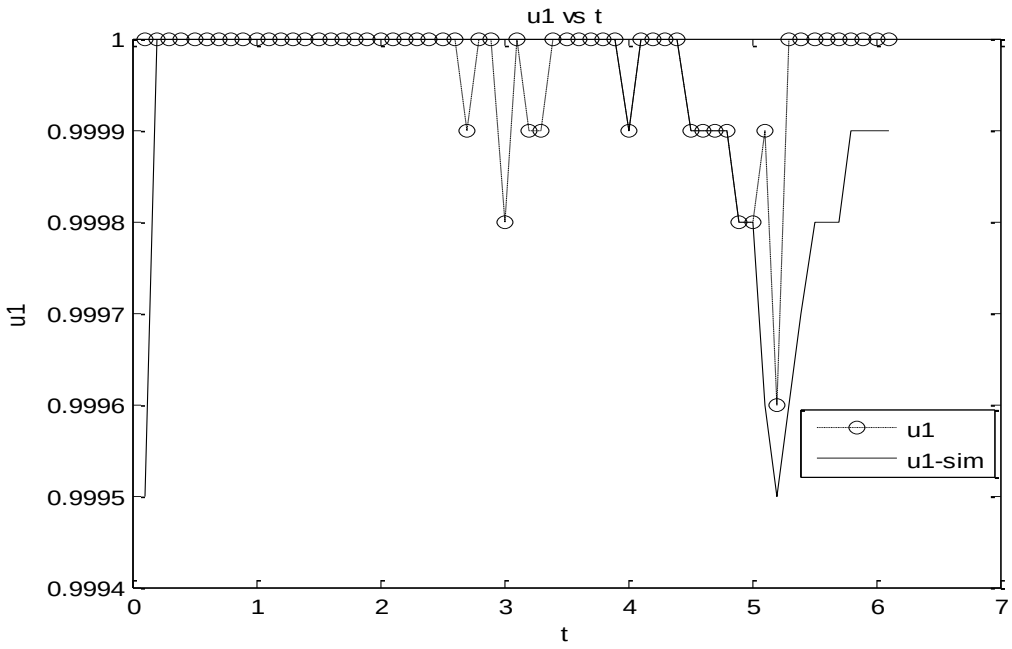
(c)



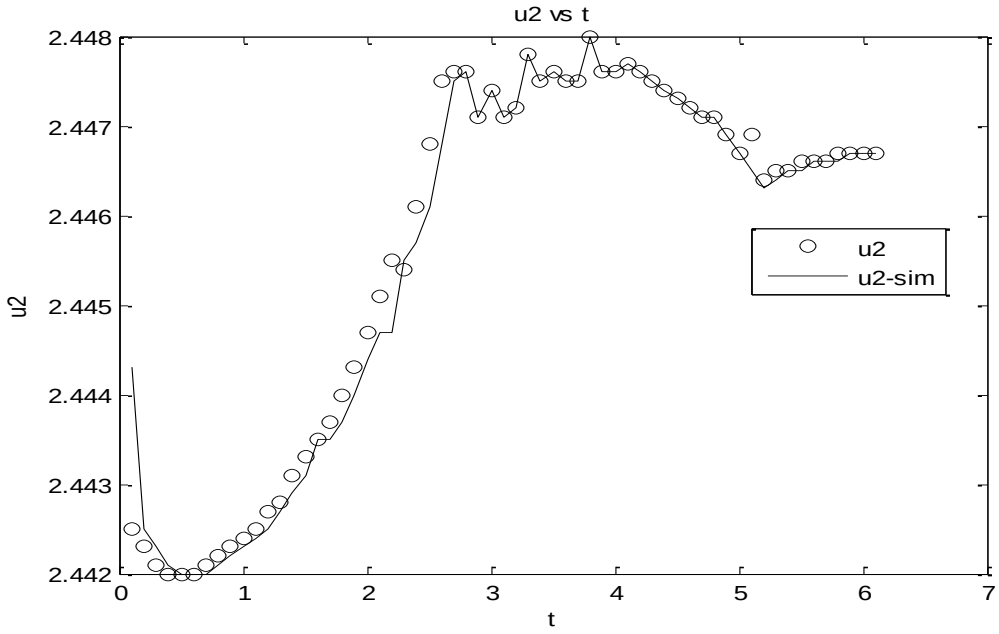
(d)

Fig. 4.1: Closed loop state (a), (b), (c) and (d) profiles for parameter adaptation.

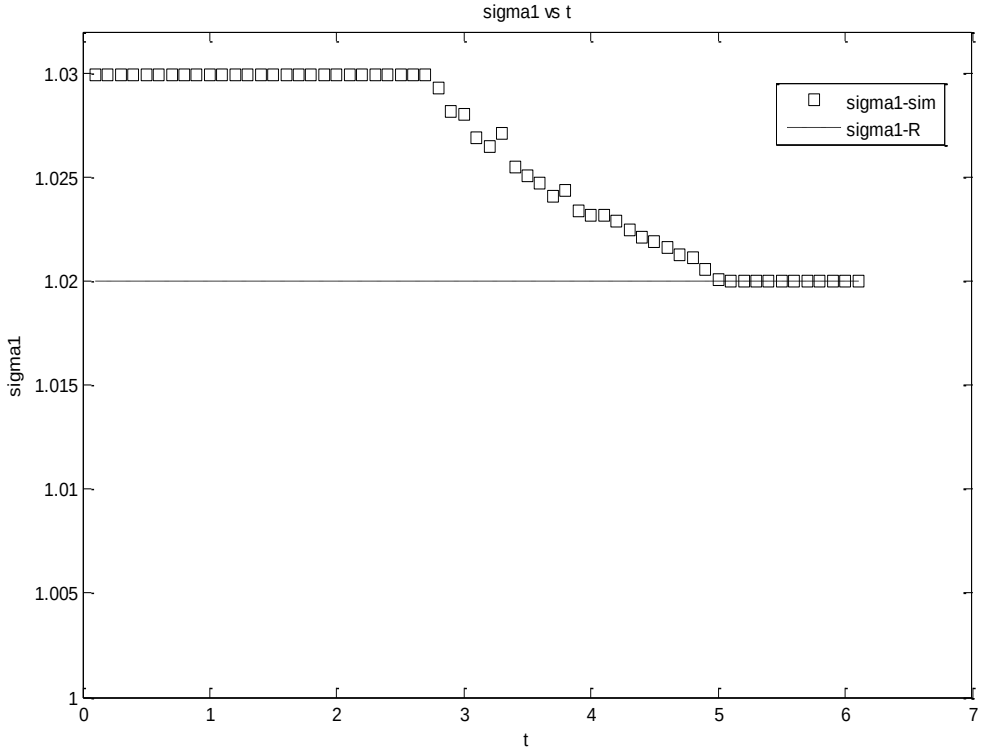
In Figure 4.2, the closed loop input trajectories, the parameter adaptation profile and the plant Lyapunov are shown. For the case of the parameter, the discontinuous line represents the true value, whereas the squares represent the simulated value.



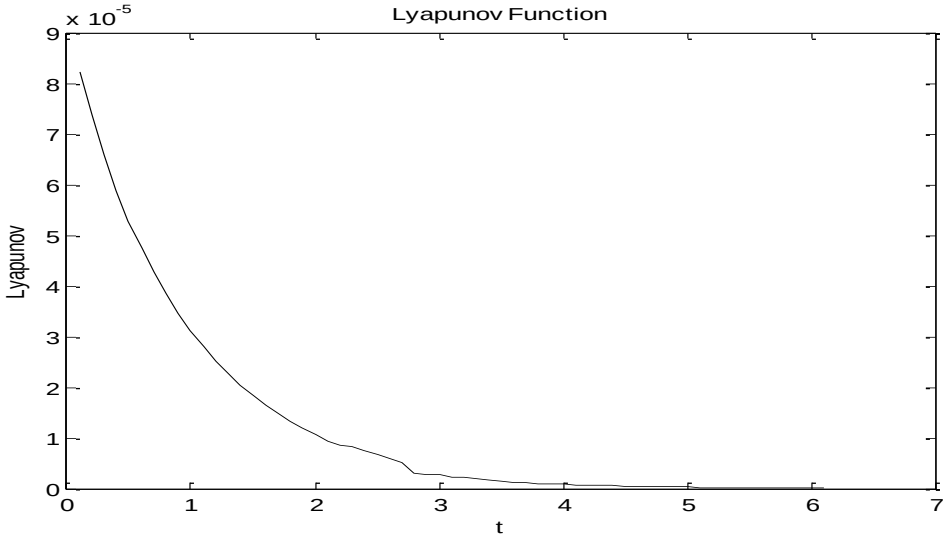
(a)



(b)



(c)



(d)

Fig. 4.2: Closed loop inputs (a) and (b), parameter adaptation (c) and Lyapunov (d) profiles.

From figure 4.2, it is evident that the optimizer chooses the upper bound for the parameter since at the initial stage it represents more profit for the cost function. The initial value for σ_1 assumed for the model was 1 and the true value in the plant was 1.02. The optimizer computes the upper bound of σ_1 until approximately t = 3 at which time the set point constraint forces the optimizer to choose a smaller value of σ_1 so as to satisfy this constraint. From t = 3 and on, the set point constraint plays the most important role as it forces the optimizer to continuously decrease the value for σ_1 until the true value has been reached. It is also important to point out that there are some points where the value for σ_1 was temporarily increased with respect to the previous iteration but nevertheless the optimization problem was still feasible.

The monotonic decrease of the Lyapunov function calculated with the actual states' values corroborates that the system is stable for the actual process during the entire simulation. Additionally, these results corroborate the proposition that the real system is well approximated by the polytopic model for the purpose of robust stability.

Parameter convergence for the general formulation 3.7 can be also analytically proven by simple arguments as given by the following proposition.

Proposition 1. Let assumptions I and II hold. If the dynamic optimization problem 4.1 is considered, at steady state $\boldsymbol{\theta}_{plant} = \boldsymbol{\theta}_{model}$

Proof.
Considering the steady state equation, $0 = f(\boldsymbol{x}s(k), \boldsymbol{u}s(k), \boldsymbol{\theta}(k))$, it can be equivalently written as:

$$\boldsymbol{x}s(k) = g(\boldsymbol{u}s(k), \boldsymbol{\theta}(k)) \qquad (4.3)$$

Then, the dynamic equation can be written as:

$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{x}s(k), \boldsymbol{K}) \qquad (4.4)$$

The deviation variables at the next time step are as follows:

$$\boldsymbol{x}'(k+1) = \boldsymbol{x}(k+1) - \boldsymbol{x}s(k+1) = (\boldsymbol{A}_0 + \Delta\boldsymbol{A})\big(\boldsymbol{x}(k) - \boldsymbol{x}s(k)\big)$$

(4.5)

$Where\ \boldsymbol{\theta}_{min} \leq \boldsymbol{\theta}(k) \leq \boldsymbol{\theta}_{max}$

The Lyapunov function for the plant model is:

$$\boldsymbol{x}'(k)^T(\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P})\boldsymbol{x}'(k) < 0 \qquad (4.6)$$

$Where\ \boldsymbol{A} = (\boldsymbol{A}_0 + \Delta\boldsymbol{A}) = \mathbb{A}_{\mathbb{T}} = \{\boldsymbol{A_{CL1}}(\boldsymbol{\theta}, k), \boldsymbol{A_{CL2}}(\boldsymbol{\theta}, k) \ldots \boldsymbol{A_{CLM}}(\boldsymbol{\theta}, k)\}.$

The "Set Point Constraint" at k+1 is:

$$\boldsymbol{x}'(k+1)^T\boldsymbol{P}\boldsymbol{x}'(k+1) < \alpha(\boldsymbol{x}'(k)^T\boldsymbol{P}\boldsymbol{x}'(k)) \qquad (4.7)$$

If should be noticed that 4.6 is always feasible, if 4.7 was feasible in the previous iteration. Feasibility of 4.7 comes from letting the set point constant for two consecutive interval times.

If $\boldsymbol{x}s'(k+1) = \boldsymbol{x}s'(k)$, then equation 4.6 equals to equation 4.7

At steady state

$$\boldsymbol{x}(k) = \boldsymbol{x}(k+1) = \boldsymbol{x}s(k) = \boldsymbol{x}measure$$

Since $\boldsymbol{x}measure$ is the first value in the prediction vector then $\boldsymbol{\theta}_{model} = \boldsymbol{\theta}_{plant}$, this is the only way by which the measured value will equate the model prediction assuming that there is no model structure error.

On the other hand the convergence of **x**s(k) to a constant value is ensured by the set point constraint (Constraint II in section 3.2.2). According to this constraint when the deviations converge to zero then the LHS of Constraint II must also be zero thus not allowing the optimizer to produce changes in the set point **x**s(k+1).

4.3 WILLIAMS-OTTO REACTOR

The Williams-Otto reactor consists of a non-isothermal CSTR with three parallel reactions.

$$\begin{aligned} A + B &\longrightarrow C && (r_1)\\ B + C &\longrightarrow P + E && (r_2)\\ C + P &\longrightarrow G && (r_3) \end{aligned}$$

The mass balances are derived from standard modeling assumptions and it is the following:

$$W\frac{dx_A}{dt} = F_A - (F_A + F_B)x_A - r_1 \qquad (4.8)$$

$$W\frac{dx_B}{dt} = F_B - (F_A + F_B)x_B - r_1 - r_2 \qquad (4.9)$$

$$W\frac{dx_C}{dt} = -(F_A + F_B)x_C + 2r_1 - 2r_2 - r_3 \qquad (4.10)$$

$$W\frac{dx_E}{dt} = -(F_A + F_B)x_E + 2r_2 \qquad (4.11)$$

$$W\frac{dx_G}{dt} = -(F_A + F_B)x_G + 1.5r_3 \qquad (4.12)$$

$$W\frac{dx_P}{dt} = -(F_A + F_B)x_P + r_2 - 0.5r_3 \qquad (4.13)$$

Where $x_A, x_B, x_C, x_E, x_G, x_P$ represent the mass fraction of the components in the reactions which are defined as: $r_1 = k_1 x_A x_B W$, $r_2 = k_2 x_B x_C W$ and $r_3 = k_3 x_C x_P W$. The reaction constants follow the standard Arrhenius equation as a function of the reactor temperature.

F_A, F_B and W represent the flow rate of A , B and mass hold up respectively.

In this problem, the manipulated inputs are the flow rate of B (F_B) and the temperature of the reactor (T_R) which is explicitly used in the exponential term of the Arrhenius equation. F_A and W are constant values. F_A can be used as the source of disturbances. The numerical values for F_A, W and k_i where taken from Amrit (2011)

The stage cost was also taken from Amrit (2011), and it is the profit of the process. Calculated from the difference between the sales of the products E and P and the costs of raw materials A and B.

$$l_P = 5554.1(F_A + F_B)x_P + 125.91(F_A + F_B)x_E - 370.3F_A - 555.42F_B \quad (4.14)$$

The inputs are subject to regular upper and lower bounds:

$$2 \leq F_B \leq 10 \quad (4.15)$$
$$349 \leq T_R \leq 367 \quad (4.16)$$

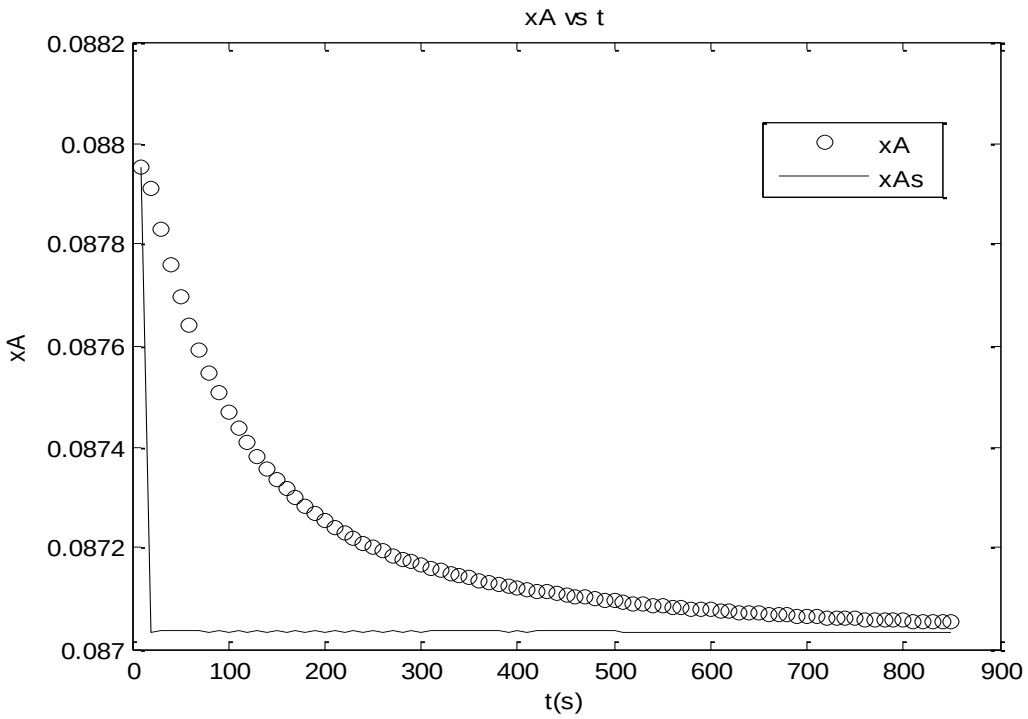
Following 3.3, the dynamic optimization problem can be stated as:

$$\min_{\boldsymbol{us},\boldsymbol{K},\boldsymbol{P}} \sum_{k=1}^{N} -(l_P{}' + Q l_P s) \quad (4.17)$$

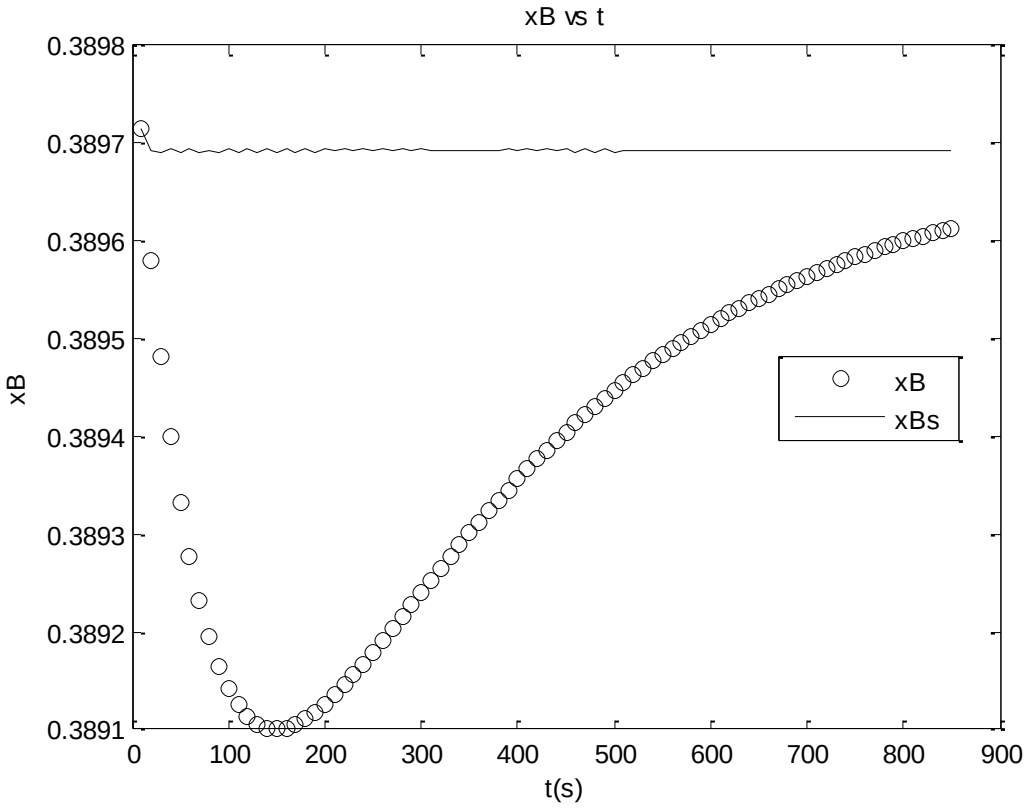
s.t.
$\boldsymbol{x}(0) = \boldsymbol{x}o$
$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k))$
$E(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{P}(k))$
$0 = f(\boldsymbol{x}s(k), \boldsymbol{u}s(k))$

Where the horizon was 15 and Q was selected to be equal to 100. Inasmuch as the static term in the stage cost is way larger than the economic term, the system is driven to the best steady state.

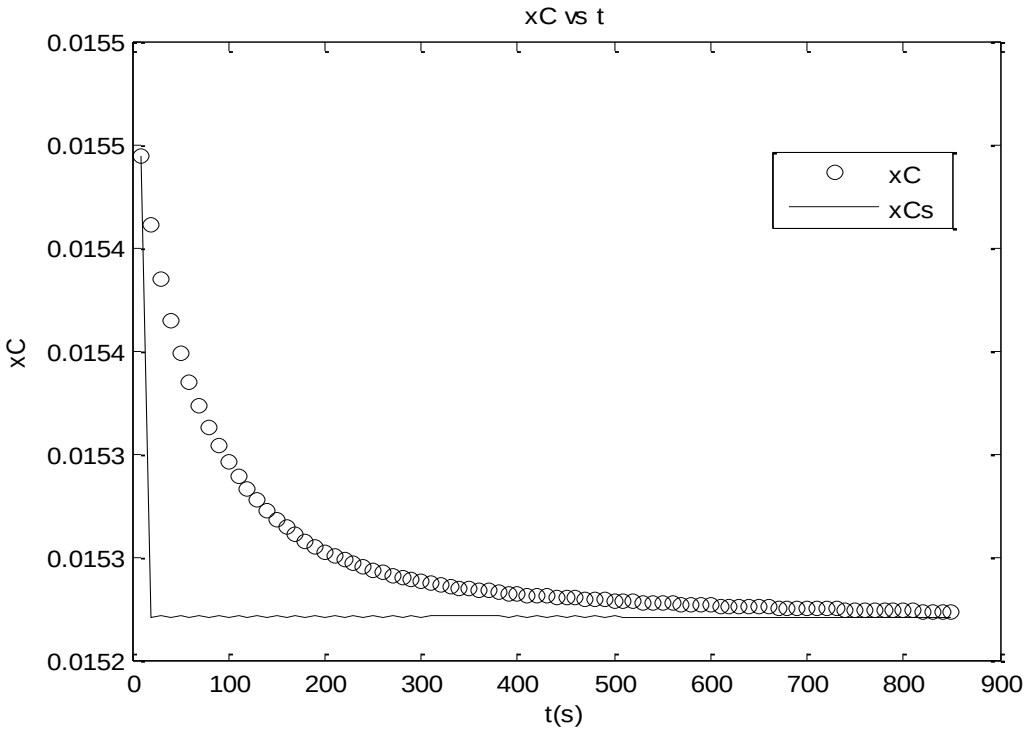
Figure 4.3 shows the closed loop state trajectories for a suboptimal operation initial condition (suboptimal steady state).


xA vs t
0.0882
0.088
0.0878
0.0876
0.0874
0.0872
0.087
xA
0
100
200
300
400
500
600
700
800
900
t(s)
xA
xAs

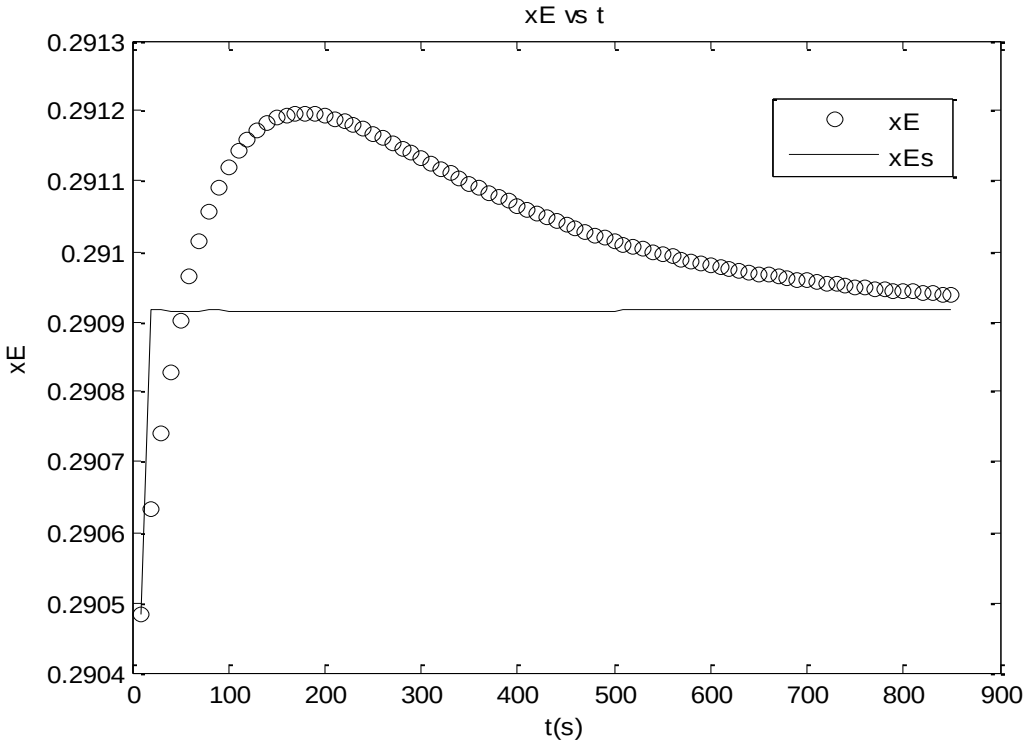
(a)


xB vs t
0.3898
0.3897
0.3896
0.3895
0.3894
0.3893
0.3892
0.3891
xB
0
100
200
300
400
500
600
700
800
900
t(s)
xB
xBs

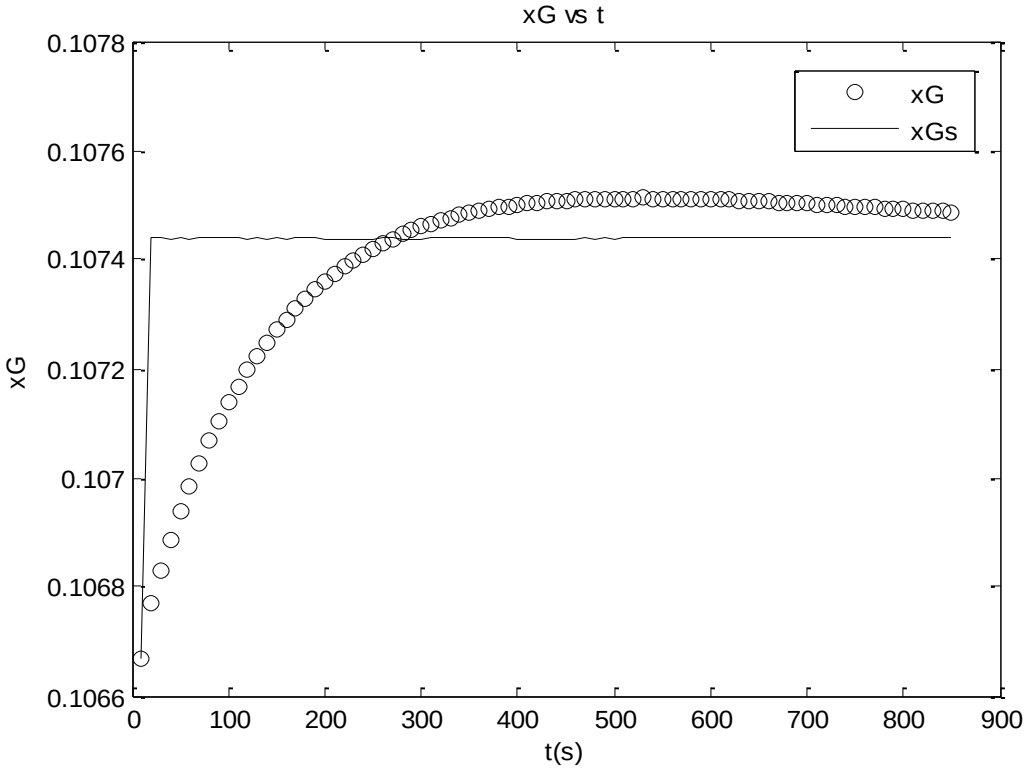
(b)


xC vs t
0.0155
0.0155
0.0154
0.0154
0.0153
0.0153
0.0152
xC
0
100
200
300
400
500
600
700
800
900
t(s)
xC
xCs

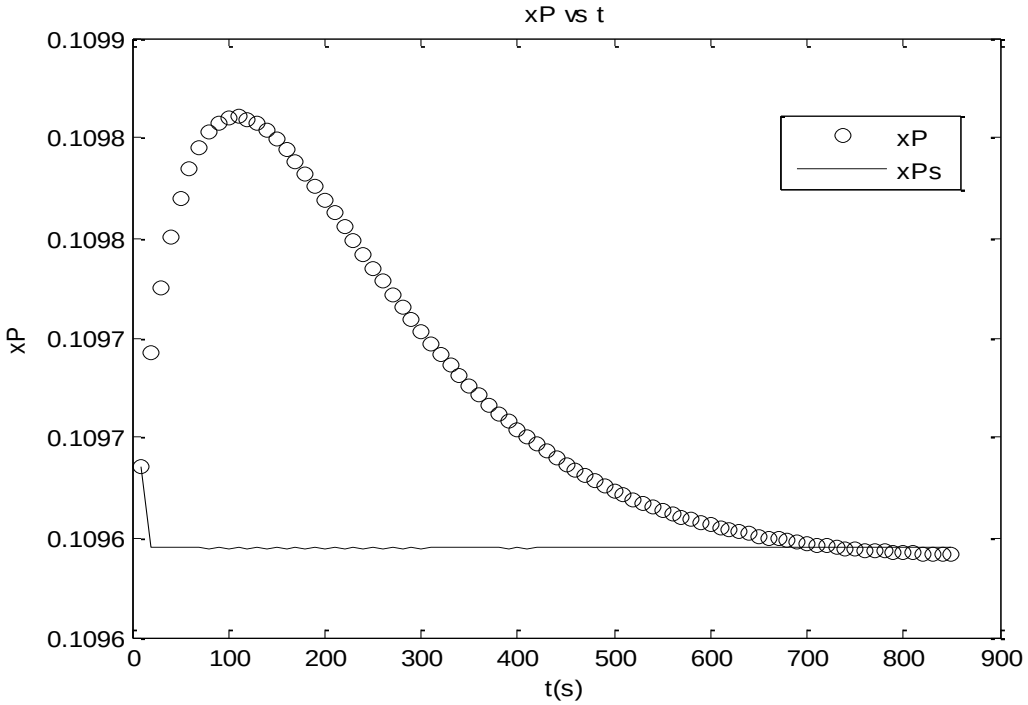
(c)


xE vs t
0.2913
0.2912
0.2911
0.291
0.2909
0.2908
0.2907
0.2906
0.2905
0.2904
xE
0
100
200
300
400
500
600
700
800
900
t(s)
xE
xEs

(d)



(e)

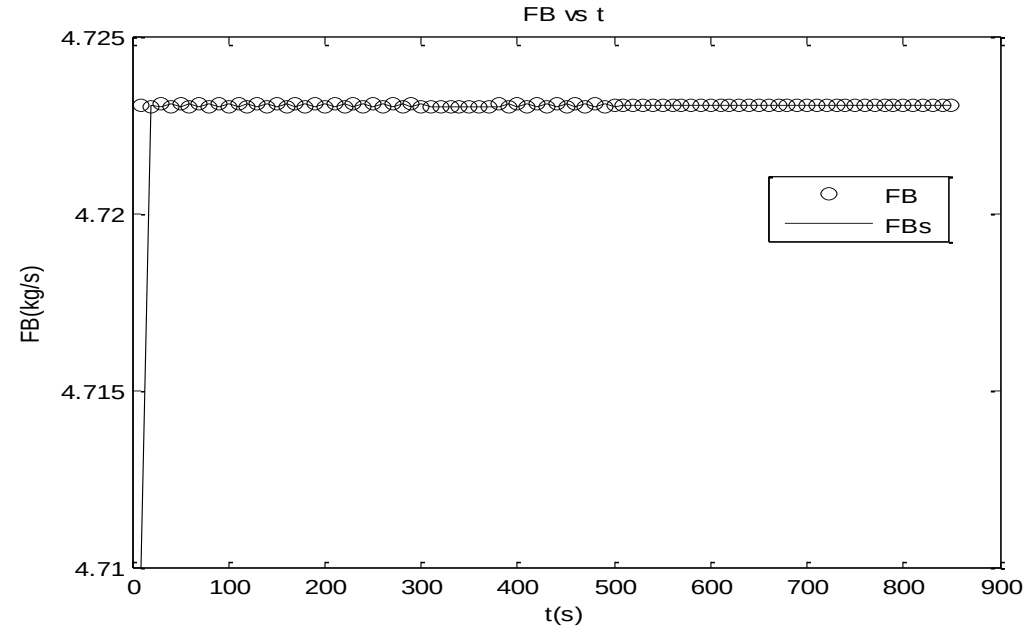


(f)

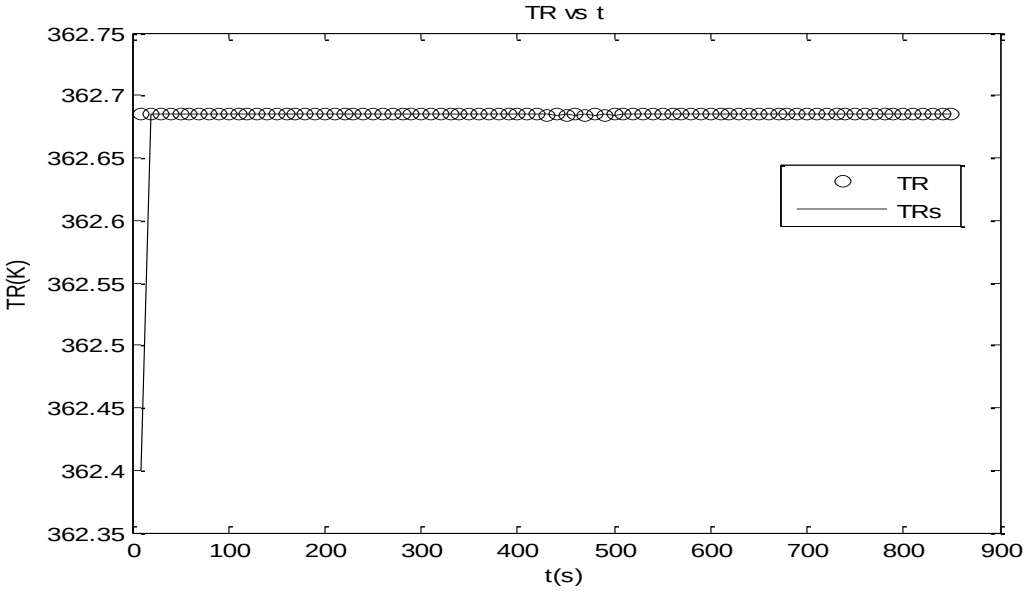
Fig. 4.3: Closed loop state (a), (b), (c), (d), (e) and (f) profiles for Otto reactor.

From this figure, it is seen that x_E, x_G, and x_P exhibit an underdamped behavior that quickly settles down at the steady state. Also it is evident that the plant is steered to the computed best steady state.

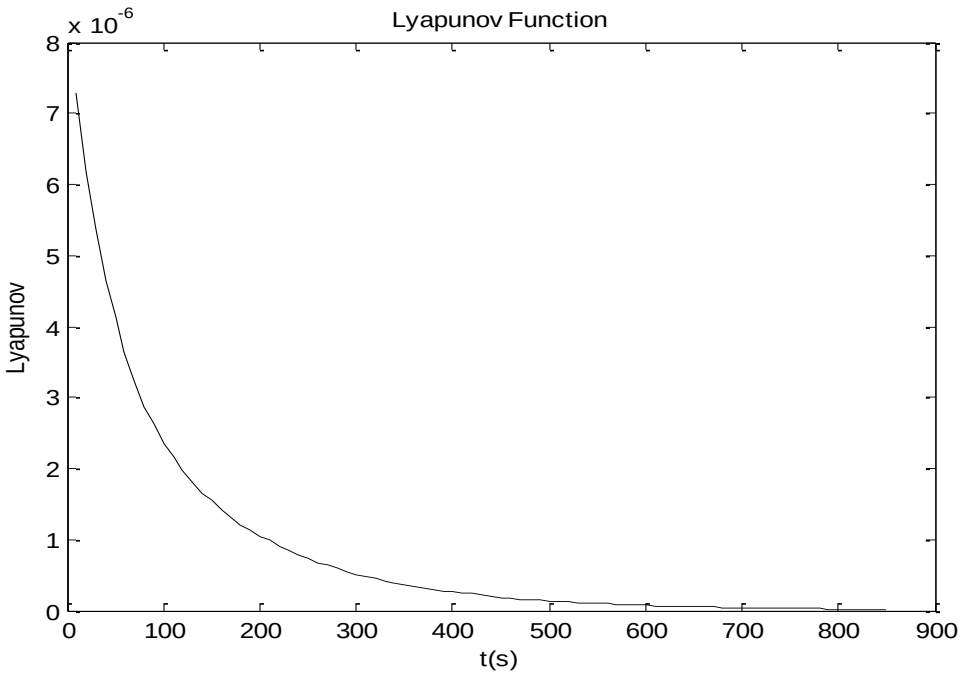
Figure 4.4 shows the closed loop inputs trajectories, the Lyapunov and profit (l_P) profiles.



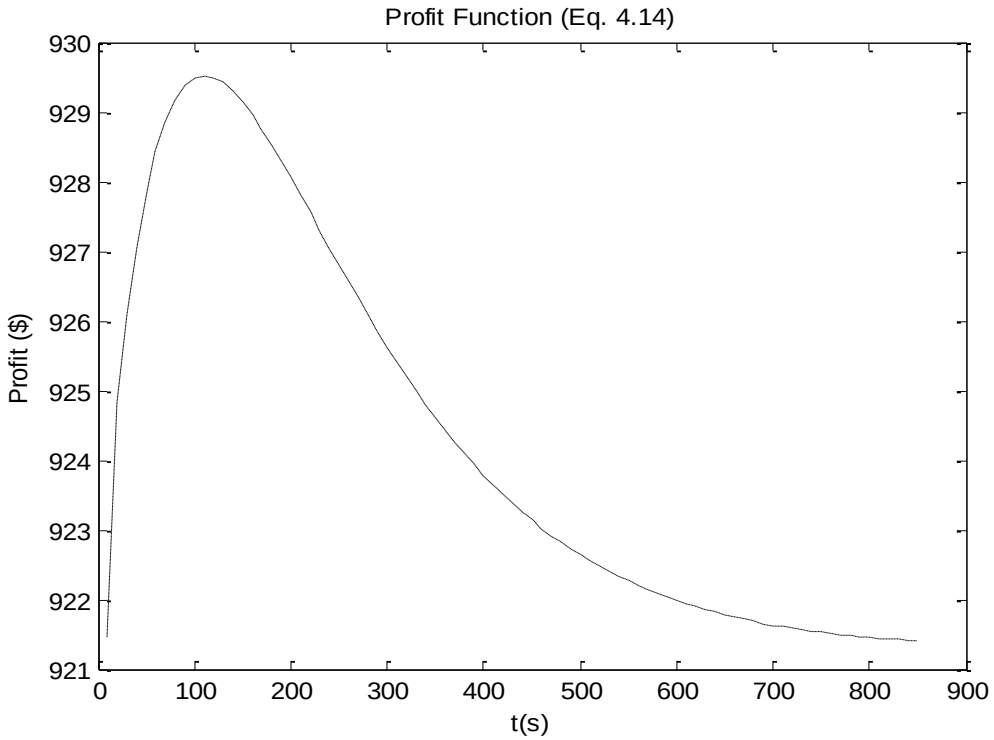
(a)



(b)



(c)



(d)

Fig. 4.4: Closed loop input (a) and (b) trajectories, Lyapunov (c) and Profit (d) profiles for Otto reactor.

This figure shows that the plant is stable and the Lyapunov function in (c) is monotonically decreasing. Inasmuch as the Lyapunov function strictly decreasing, the initial condition is significant. Furthermore, the profit also exhibits an underdamped behavior as shown in figure 4.4. This is expected since the cost is a direct function of the states that also exhibit underdamped convergence. It is also shown that the profit is settling down at the best steady state (cost=921.5169) outperforming the suboptimal steady state initial condition. The transient behavior for the entire simulation produced an improvement in the profit of 244.74 $ compare to the suboptimal initial condition.

The closed loop inputs in figure 4.4 show that practically there is not a deviation between the best steady state and the manipulated variables. This is because Q was selected to provide the best steady state implying that $Ql_Ps \gg l_P'$. Furthermore, in contrast to the example in 3.3 the manipulated variables at the best steady state in this case is not at any bound.

The computed best steady state by the proposed EMPC algorithm is:

$\boldsymbol{xs} = [0.0870, 0.3896, 0.0152, 0.2909, 0.1074, 0.1096]^T$

$\boldsymbol{us} = [4.7231, 362.6852]^T$

Which equals the best steady state computed off-line.

4.4 SOLUTION OF DYNAMIC OPTIMIZATION PROBLEM

This section discusses the feasibility of the EMPC approach and different improvements to the numerical solution.

4.4.1 Feasibility

Feasibility of the algorithm implies that the algorithm complies with all the constraints including the four stability related constrains introduced in 3.2.

Next, it is explained how one can ensure feasibility of these constraints.

1. “Set Point Constraint”

The Set Point constraint is satisfied because for the special case that **x**s(k) remains fixed at the value from the previous iteration, then the set point constraint (Constraint II in section 3.2) becomes equal to the robust Lyapunov stability condition (constraint IV in section 3.2).

This become obvious from 3.10

$$\begin{aligned}&\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big)^T\boldsymbol{P}(k-1)\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big)\\&\qquad<\alpha\big(\boldsymbol{x}(k-1)-\boldsymbol{x}s(k-1)\big)^T\boldsymbol{P}(k-1)\big(\boldsymbol{x}(k-1)-\boldsymbol{x}s(k-1)\big)\end{aligned}$$

If the Lyapunov function is strictly decreasing the set point constraint is feasible.

2. “ P constraint”

The P constraint is satisfied by fixing the **P** matix from the previous iteration.

From 3.11

$$\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big)^T\boldsymbol{P}(k)\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big)\leq\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big)^T\boldsymbol{P}(k-1)\big(\boldsymbol{x}(k)-\boldsymbol{x}s(k)\big)$$

It is seen that if the **P** in the LHS and RHS are the same, the constraint is satisfied.

3. “Positive definite **P**”

This constraint is satisfied by following the same procedure as 2. Since the fact that **P** from the previous iteration is positive define the P at time (k) will also be positive definite.

4. “Lyapunov Based Robust Stability Constraint”

For open loop stable systems, this constraint can be automatically satisfied by making **K** in equation 3.15 equal to cero. Therefore, this is reduced to:

$$\boldsymbol{A_{CL}}(\boldsymbol{\theta}, k) = \boldsymbol{A}(\boldsymbol{\theta}) \qquad (4.18)$$

If the system is open loop stable, the inequality in 3.19

$$V(k+1) - V(k) < 0$$

is satisfied. For systems that are open loop unstable one option is to stabilize them by using a proportional only controller and then apply the proposed methodology to the stabilized system. Another possibility to ensure feasibility for both open and closed loop systems is to formulate a series of LMI inequalities that are solved off-line for the worst case scenario. The investigation of this option is left for future research.

A logical flowchart to ensure feasibility of the system based on the arguments given in the current section is schematically described in the figure 4.5.

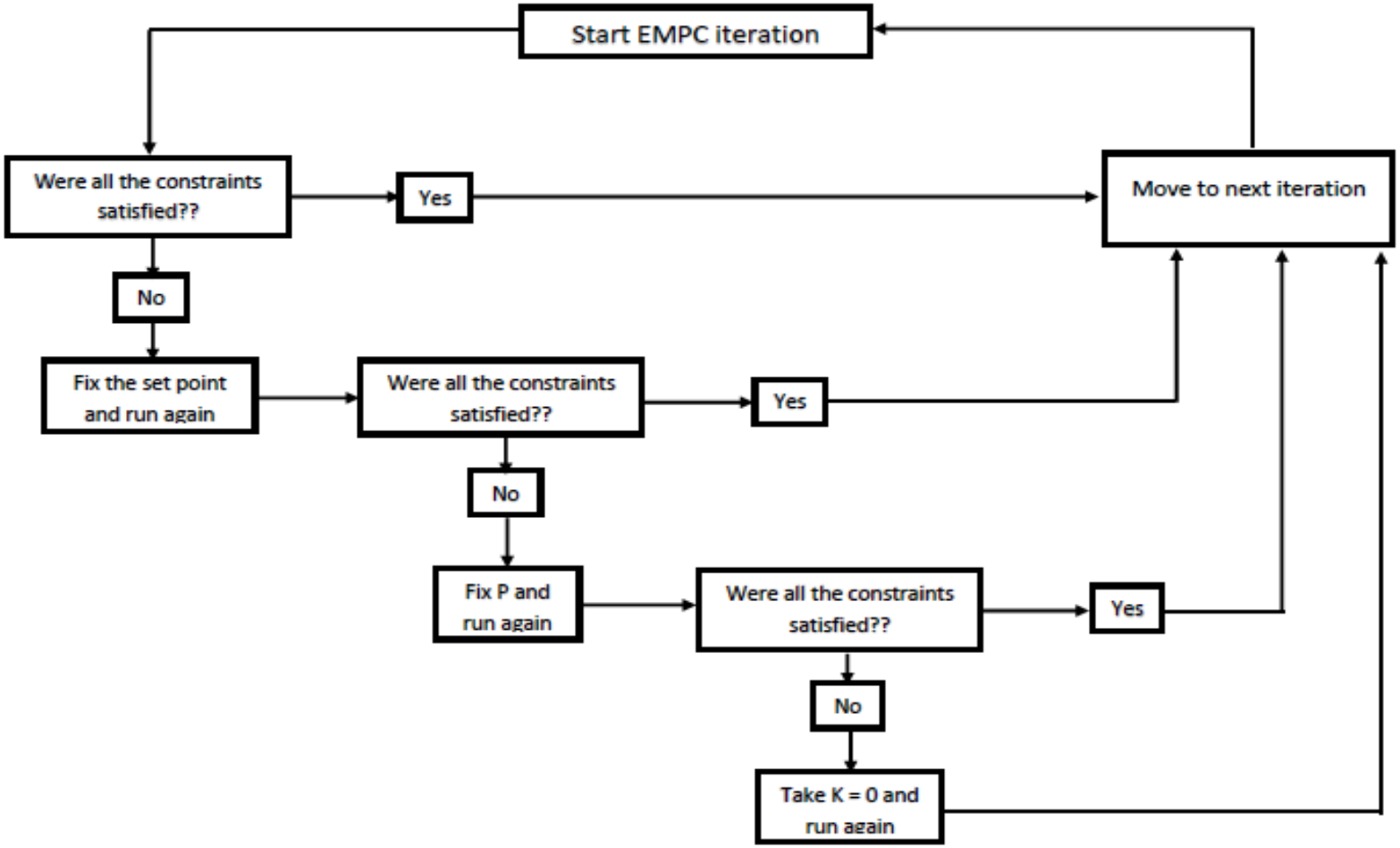


Fig. 4.5: Diagram showing the implementation of the feasibility problem

4.4.2 Numerical Solution

As the proposed EMPC is applied online, fast numerical solution of the dynamic optimization problem is essential for real time implementation. For real applications the solution of these optimization problems should been completed faster than the sampling interval. In the present work, we are solving the Economic Model Predictive Controllers using modest computing power (Intel i5 2.5 GHz) and commercial software (Matlab) and thus the potential for improvements are limited. However there are still some actions that can be taken in order to enhance the speed of the solution. The implementation of these steps may speed up the solution but at the cost of a reduction of degrees of freedom available for optimization.

- Manipulated Variables.

  Considering equation 3.3

$$\boldsymbol{u' = Kx}$$

  In order to reduce the number of manipulated variables in the matrix, each row is made of equal elements.

$$\begin{pmatrix} k_1 & \cdots & k_1 \\ \vdots & \ddots & \vdots \\ k_n & \cdots & k_n \end{pmatrix} \tag{4.19}$$

  Thus, instead of a total of nxm variables (where n is the number of rows and m the number of columns), the number of decision variables is reduced to n.

  Furthermore, since equation 3.3 is solved for the entire horizon *N*, it is possible to arrange the manipulated variables into blocks where the manipulated variables are kept constant within each block thus reducing even more the number of manipulated variables.

For example if N = 10, for the matrix $\boldsymbol{K}$ can be divided into 3 blocks corresponding to 3 different periods of time along the prediction horizon:

$$\boldsymbol{K} = \begin{cases} \boldsymbol{K_1}, & N \leq 3 \\ \boldsymbol{K_2}, & 3 < N \leq 7 \\ \boldsymbol{K_3}, & 7 < N \leq 10 \end{cases} \tag{4.20}$$

- Positive Definite Matrix "**P**"

The largest number of decision variables in the examples discussed in this thesis originated from the elements of the matrix **P.** With the addition of any new state **P** grows quadratically, for instance, a $\boldsymbol{P_{4x4}}$ has sixteen elements whereas $\boldsymbol{P_{5x5}}$ has twenty five.

For small size problems involving less than 10 states, and taking advantage of the definition of a positive definite matrix, the following decomposition can be performed for reducing the number of optimization variables:

$$\begin{pmatrix} P_1 & 0 & \dots & 0 \\ 0 & P_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & P_n \end{pmatrix} + \begin{pmatrix} 0 & P_{n+1} & & \dots & P_{2n} \\ & 0 & 0 & P_{2n+1} & \vdots \\ & & \vdots & 0 & \ddots & P_{n^2-\sum_{i=1}^{n-1} i} \\ & & 0 & \cdots & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & P_{n+1} & & \dots & P_{2n} \\ & 0 & 0 & P_{2n+1} & \vdots \\ & & \vdots & 0 & \ddots & P_{n^2-\sum_{i=1}^{n-1} i} \\ & & 0 & \cdots & 0 & 0 \end{pmatrix}^T = \boldsymbol{P} \tag{4.21}$$

Instead of having n^2 variables we have $n^2 - \sum_{i=1}^{n-1} i$.

For large problems, it is proposed to use the diagonal matrix in 4.20

$$\begin{pmatrix} P_1 & 0 & \dots & 0 \\ 0 & P_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & P_n \end{pmatrix} = \boldsymbol{P} \tag{4.22}$$

And enforce that each element of (4.21) is positive. Consequently, the number of optimization variables would be reduced to n

Besides these suggestions other improvements in the speed can be made by using more sophisticated optimizers, ODE solvers and programing languages. However, these potential numerical improvements are left for future research. Moreover, other more numerically effective techniques for testing positive definiteness of a matrix could be also studied instead of computing eigenvalues that is computationally intensive.

Chapter 5

Conclusions and Future work

5.1 CONCLUSIONS

Stability and EMPC:

An Economic Model Predictive Control online algorithm that can be applied not only to generic nonlinear systems but also to generic stage costs was successfully developed in Chapter 3. By adding an additional set of specific constraints, the asymptotic stability of the algorithm was numerically enforced at each time interval. The robustness property of this algorithm comes from the satisfaction of the "Lyapunov Based Robust Stability Constraint". The "Set Point Constraint" and "P Constraint" regulate the contraction of the Lyapunov function by varying the set point or **P** matrix respectively.

Applications:

The proposed EMPC algorithm was effectively applied to small and medium size examples in chapters number three and four respectively. Given the inability to converge to the best economic steady state due to the finite small horizon, the stage cost was splitted into a dynamic cost and a steady state cost. The latter promoted convergence to the steady state. A weight multiplying the steady state cost term was used as a tuning parameter. Although, a formal methodology for choosing this weight has not been provided in this thesis, it was argued that the choice could be possibly related to the frequency of disturbances entering the process that may require operating at steady state or at transient state. If the frequency of the disturbances is high or if best steady state operation is not required, Q could be chosen so that the performance on average outperforms or is as good as the best economic steady state.

Furthermore, it was shown that the proposed EMPC algorithm has some advantages compared to previously reported EMPC algorithm that served as motivation for this work. In contrast with the study of Rawlings et al, it was shown that the proposed algorithm bypasses several of the

requirements and potential limitations of Rawlings' algorithm. For instance, the current algorithm does not require off line computations of the best economic steady state and it does not need terminal constraints, terminal costs or/and addition of convex terms to the stage cost. Potentially, terminal constraints or terminal costs may limit the closed loop performance and the addition of tracking terms to the cost function may force the system to track closely a steady state which may not be optimal during transients. On the other hand, the algorithm has some drawbacks such as high computational requirements since robust stability is numerically enforced on-line and feasibility of the algorithm can only be ensured for open loop stable systems only. Some options were suggested for open loop unstable systems such as stabilization by a purely proportional controller.

Parameter adaptation, feasibility and solution:

The proposed algorithm was also formulated to perform one of the most important tasks of RTO strategies namely adaptation of model parameters. In chapter number four, it was demonstrated that by means of the "Set Point Constraint" and using the parameter as an optimization variable, the model parameter was adapted in a finite time. Furthermore, the adaptation rate can be increased by decreasing the value of the factor α in the RHS of the set point constraint. Attention must be put in not making the optimization problem infeasible due to a small value of α. The "Set Point Constraint" plays a fundamental role in the adaptation problem inasmuch as it forces the system to reach the true value of the parameter. In terms of feasibility, it was proven that the system is always feasible for open loop stable systems. By fixing the set point and **P** matrix, the respective constraints can be satisfied. By making **K** = 0 the Lyapunov Based Robust Stability Constraint can be satisfied. It was also proposed to speed up the solution by decomposing the **P** matrix and using less optimization variables for the control gain matrix **K.**

5.2 FUTURE WORK

Solution of higher dimensional problems.

In this work the EMPC algorithm was only applied to relatively small size problems with less than ten states. The proposed methodology should be study for nonlinear systems with much larger number of states and manipulated variables since chemical processes are generally described by large systems of equations. Special attention must be put on the efficient solution of

the positive definite property of **P** since it is the constraint that has the most number of optimization variables and it has the most effect on computational requirements.

Use of a better optimization software.

For the solution of the dynamic optimization problems the fmicon optimization function of Matlab was used. As reported in literature, in terms of speed and performance, this may not be the best or fastest option. Consequently, the use of better optimizers and ODE solvers must be explored along with special techniques for solving DAE systems such as direct methods. Indeed, in the literature, most nonlinear dynamic optimization problems even for small cases are solved with Ipopt. Therefore, the implementation of this kind of optimisers should be considered.

Open loop unstable systems.

One of the most important drawback of this algorithm is that it can only be applied to open loop stable systems. In real applications, there are situations where open loop unstable systems will have to be considered. Then, necessary modifications have to be studied. One possibility is to ensure feasibility a priori by means of a series of linear matrix inequalities to account for the worst case scenario.

Systematic selection of tuning parameters (Q and α)

Although some inside was given for the systematic selection of not only Q but α, limited work was performed. Simulations demonstrating the influence in the performance related to the selection of these two tuning parameters must be carried out.

REFERENCES

M. Ellis, H. Durand, P.D. Christofidies, A tutorial review of economic model predictive control methods, J. Process Control 24 (2014) 1156-1178.

J.B. Rawlings, D. Angeli, C. N. Bates, Fundamentals of economic model predictive control, 51st IEEE Conference on Decision and Control, December 10-13. 2012. Maui, Hawaii, USA.

D. Angeli, R. Amrit, J.B. Rawlings, On average performance and stability of economic model predictive control, IEEE Trans. Automat. Control 57 (2012), 1615-1626.

R. Amrit, J.B. Rawlings, L.T. Biegler, Optimizing process economics online using model predictive control, Comput. Chem. Eng. 58 (2013), 334-343.

Adetola, M. Guay, Integration of real-time optimization and model predictive control, J. Process Control 20 (2010) 125-133.

M. Heidarinejad, J. Liu, P. D. Christofidies, Economic model predictive control of nonlinear process using lyapunov techninques, AlChE J. 58 (2012) 855-870.

M. Ellis, P. D. Christofidies, Economic model predictive control with time varying objective function for nonlinear process systems, AlChE J. 60 (2013) 507-519.

M. Ellis, P. D. Christofidies, Performance monitoring of economic model predictive control systems, Ind. Eng. Chem. Res. 53 (2014) 15406-15413.

M. Ellis, P. D. Christofidies, Integrating dynamic economic optimization and model predictive control for optimal operation of nonlinear process systems, Control Eng. Pract. 22 (2014) 242-251.

M. Diehl, R. Amrit, J.B. Rawlings, Lyapunov function for economic optimizing model predictive control, IEEE Trans. Automat. Control 56 (2011), 703-707.

R. Amrit, J. B. Rawlings, D. Angeli, Economic optimization using model predictive control with a terminal cost, Annual Reviews in Control. 35 (2011) 178-186.

D. Angeli, R. Amrit, J. B. Rawlings, Receding horizon cost optimization for overly constrained nonlinear plants, 48st IEEE Conference on Decision and Control, December 16-18. 2009. Shanghai, China.

D. Angeli, R. Amrit, J. B. Rawlings, Enforcing convergence in nonlinear economic MPC, 50st IEEE Conference on Decision and Control, December 12-15. 2011. Orlando, FL, USA.

R. Amrit, Optimizing process economics in model predictive control, PhD Thesis. University of Wisconsin-Madison. 2011.

L.T. Biegler, Nonlinear programming. Concepts, algorithms, and application to chemical processes. Philadelphia, PA, USA: SIAM 2010.

T. Binder, L. Blank, H. Bock, R. Bulirsch, W. Dahmen, M. Dielh, Introduction to model based optimization of chemical processes on moving horizons online optimization of large scale systems: State of the art. Berlin Heidelberg, Germany: Springer (2001).

C. K. Lee, J. E. Bailey, Modification of consecutive competitive reaction selectivity by periodic operation, Ind. Eng. Chem. Process Des. 19 (1980) 160 -166.

Gopalakrishnan, L.T. Biegler, Economic nonlinear model predictive control for periodic optimal operation of gas pipeline networks, Computers and Chemical Engineering 52 (2013) 90-99.

W. Al-Gherwi, Robust distributed model predictive control strategies of chemical processes, PhD Thesis, University of Waterloo. 2010.

R. Huang, E. Harinath, L.T. Biegler, Lyapunov stability of economically oriented NMPC for cyclic processes, J. Process Control 21 (2011) 501-509.

R. Huang, L.T. Biegler, E. Harinath, Robust stability of economically oriented infinite horizon NMPC that include cyclic processes, J. Process Control 22 (2012) 51-59.

J. Ma, S. J. Qin, T. Salsbury, Application of economic mpc to energy and demand minimization of a commercial building, J. Process Control 24 (2014) 1282-1291.

E. A. N. Idris, S. Engell, Economics-based nmpc strategies for the operation and control of a continuous catalytic distillation process, J. Process Control 22 (2012) 1832-1843.

L. Grune, J. Pannek, Nonlinear Model Predictive Control: Theory and Algorithms. Springer-Verlag London, (2011).

M. Mahmoud, Advances in Discrete Time Systems. InTech, (2012).

Appendix A

Routine for the solution of EMPC. EMPC is solved as a dynamic optimization problem in a receding horizon basis (feedback).

Appropriate modifications must be made to the steady state model, cost function, dynamic model, constraints, inputs etc. (subroutines) to solve the all the problems described in this work. The subroutines shown here are for the section 3.3. This code was modified from Grune and Pannek (2011).

Template with initial conditions and function to call

```
mpciterations = 2;
N             = 15;
T             = 0.1;
Pmeasure      = [0.4328    0.6360    0.3574    0.3644    1.4253    0.5502
0.5703 0.5263    0.4408    0.7601];
Ameasure      = 0;
Neromeasure   = -1;
Constmeasure  = -1;
Constmeasure2 = -1;
tmeasure      = 0.0;
xmeasure      = [0.3845 1.5850 0.3730 0.2365 0.3845 1.5850 0.3730 0.2365
0.994 2.4310 1];
u0            =
[0*ones(1,N);0*ones(1,N);0.999*ones(1,N);2.431*ones(1,N);[Pmeasure',ones(10,N
-1)];ones(1,N)];
tol_opt       = 1e-6;
opt_option    = 1;
iprint        = 0;
type          = 'differential equation';
atol_ode_real = 1e-6;
rtol_ode_real = 1e-6;
atol_ode_sim  = 1e-6;
rtol_ode_sim  = 1e-6;
```

```
[t,x,u,Res,L,BB,U] = nmpc(@runningcosts, @terminalcosts,
@constraints,@terminalconstraints, @linearconstraints,
@contsystem_ct,mpciterations,@avconstraint,@evaluaciones,@inicializa,@valores
xs,@constraintstotales4,@constraintstotales2,@evalP,Pmeasure,@obtencionP,Amea
sure,Neromeasure,Constmeasure,Constmeasure2,@obtencionA,@proceso2,@computeu,
N, T, tmeasure, xmeasure, u0, tol_opt, opt_option, type, atol_ode_real,
rtol_ode_real, atol_ode_sim, rtol_ode_sim);
```

Economic Model Predictive Control Algorithm and settings for the optimizer. Note that SQP was chosen for all the simulations.

```
function [t, x, u, Res, LL, BB, U] = nmpc(runningcosts, terminalcosts, ...
constraints, terminalconstraints, ...
linearconstraints, system, ...
mpciterations,
avconstraint,system3,system4,system5,constraintstotales,constraintstotales2,s
ystem6,Pmeasure,obtencionP,Ameasure,Neromeasure,Constmeasure,Constmeasure2,ob
tencionA,proceso2,computeu, N, T, tmeasure, xmeasure, u0, ...
varargin)

global JJJ



    if (nargin>=29)
        tol_opt = varargin{1};
    else
        tol_opt = 1e-6;
    end;
    if (nargin>=30)
        opt_option = varargin{2};
    else
        opt_option = 0;
    end;
    if (nargin>=31)
        if ( strcmp(varargin{3}, 'difference equation') || ...
                strcmp(varargin{3}, 'differential equation') )
            type = varargin{3};
        else
            fprintf([' Wrong input for type of dynamic: use either ', ...
```

```
                '"difference equation" or "differential equation".']);
        end
    else
        type = 'difference equation';
    end;
    if (nargin>=32)
        atol_ode_real = varargin{4};
    else
        atol_ode_real = 1e-8;
    end;
    if (nargin>=33)
        rtol_ode_real = varargin{5};
    else
        rtol_ode_real = 1e-8;
    end;
    if (nargin>=34)
        atol_ode_sim = varargin{6};
    else
        atol_ode_sim = atol_ode_real;
    end;
    if (nargin>=35)
        rtol_ode_sim = varargin{7};
    else
        rtol_ode_sim = rtol_ode_real;
    end;
    if (nargin>=36)
        iprint = varargin{8};
    else
        iprint = 0;
    end;

% Determine MATLAB Version and
    % specify and configure optimization method
    vs = version('-release');
    vyear = str2num(vs(1:4));
    if (vyear <= 2007)
        fprintf('MATLAB version R2007 or earlier detected\n');
```

```
    if ( opt_option == 0 )
        options = optimset('Display','off',...
            'TolFun', tol_opt,...
            'MaxIter', 2000,...
            'LargeScale', 'off',...
            'RelLineSrchBnd', [],...
            'RelLineSrchBndDuration', 1);
    elseif ( opt_option == 1 )
        error('nmpc:WrongArgument', '%s\n%s', ...
              'Interior point method not supported in MATLAB R2007', ...
              'Please use opt_option = 0 or opt_option = 2');
    elseif ( opt_option == 2 )
         options = optimset('Display','off',...
             'TolFun', tol_opt,...
             'MaxIter', 2000,...
             'LargeScale', 'on',...
             'Hessian', 'off',...
             'MaxPCGIter', max(1,floor(size(u0,1)*size(u0,2)/2)),...
             'PrecondBandWidth', 0,...
             'TolPCG', 1e-1);
    end
else
    fprintf('MATLAB version R2008 or newer detected\n');
    if ( opt_option == 0 )
        options = optimset('Display','off',...
            'TolFun', tol_opt,...
            'MaxIter', 10000,...
            'Algorithm', 'active-set',...
            'FinDiffType', 'forward',...
            'RelLineSrchBnd', [],...
            'RelLineSrchBndDuration', 1,...
            'TolConSQP', 1e-6);
    elseif ( opt_option == 1 )
        options = optimset('Display','notify',...
            'TolFun', tol_opt,...
            'TolCon', 1e-6,...
            'MaxFunEvals', 3000,...
```

```
                'MaxIter', 4000,...
                'Algorithm', 'sqp',...
                'AlwaysHonorConstraints', 'bounds',...
                'FinDiffType', 'forward',...
                'HessFcn', [],...
                'Hessian', 'bfgs',...
                'HessMult', [],...
                'InitBarrierParam', 0.1,...
                'InitTrustRegionRadius', sqrt(size(u0,1)*size(u0,2)),...
                'MaxProjCGIter', 2*size(u0,1)*size(u0,2),...
                'ObjectiveLimit', -1e20,...
                'ScaleProblem', 'obj-and-constr',...
                'SubproblemAlgorithm', 'cg',...
                'TolProjCG', 1e-2,...
                'TolProjCGAbs', 1e-10,'UseParallel','always');
        elseif ( opt_option == 2 )
            options = optimset('Display','off',...
                'TolFun', tol_opt,...
                'MaxIter', 2000,...
                'Algorithm', 'trust-region-reflective',...
                'Hessian', 'off',...
                'MaxPCGIter', max(1,floor(size(u0,1)*size(u0,2)/2)),...
                'PrecondBandWidth', 0,...
                'TolPCG', 1e-1);
        end
    end

warning off all
    t = [];
    x = [];
    u = [];
    Res = [];
    LL = [];
    BB = [];
    U=[];
    % Start of the NMPC iteration
    mpciter = 1;
```

```
    while(mpciter < (mpciterations+1))


        if (mpciter==1)
            factor=0;
            YYZ=0;
            Sub=-1;
        end


        % Step (1) of the NMPC algorithm:
        %   Obtain new initial value
        [t0, x0, P0,A0,Nero0,Const0,Const20,factor0,YYZ0,Sub0] =
measureInitialValue ( tmeasure, xmeasure, Pmeasure,
Ameasure,Neromeasure,Constmeasure,Constmeasure2,factor,YYZ,Sub);
        % Step (2) of the NMPC algorithm:
        %   Solve the optimal control problem
        t_Start = tic;
        [u_new, V_current, exitflag, output] = solveOptimalControlProblem ...
            (runningcosts, terminalcosts, constraints, ...
            terminalconstraints, linearconstraints, system, ...

avconstraint,system3,system4,system5,constraintstotales,constraintstotales2,m
pciter,Nero0,Const0,Const20, N,P0,A0,factor0,YYZ0,Sub0,t0, x0, u0, T, ...
            atol_ode_sim, rtol_ode_sim, tol_opt, options, type);
        t_Elapsed = toc( t_Start );



        %   Store closed loop data
        TTT = [ t; tmeasure ];
        XXX = [ x; xmeasure ];
        UUU = [ u; u_new(:,1)'];



    x1b=
computeOpenloopSolution(system,system3,system4,system5,mpciter,Nero0,Const0,C
onst20,factor0,YYZ0,Sub0, N, T, t0, x0, u_new, ...
                                atol_ode_sim, rtol_ode_sim, type);


    if (Const20<0)
```

```
        if (Nero0<=0.000001)
            if (Const0>0)||(factor0==10)||(Sub0>0)
               S=0;

            else
               S=1;
            end
        else
         S=1;
        end
    else
        S=1;
    end


     if (mpciter==1)
     cnew2= constraintstotales(t0,x1b,u_new,N,T,S);
     c = [cnew2(1,1:2),-1,cnew2(1,3:6)];
     else
     cnew2=
constraintstotales2(t0,x1b,u_new,N,P0,A0,T,S,Nero0,Const20,Const0,Sub0);
     c = cnew2;
     end
         Res1 = [Res; c];


     B0=obtencionA(t0,x1b,u_new,N,Nero0,P0,Const20,Const0,Sub0);
     BB1 =[BB;B0];




         Lnew = lyapunov
(system,system3,system4,system5,obtencionP,mpciter,Nero0,Const0,Const20,P0,fa
ctor0,YYZ0,Sub0, N, T, t0, x0, u_new, ...
                                      atol_ode_sim, rtol_ode_sim, type);


         LL1=[LL;Lnew];


         if (mpciter>1)
```

```
            if (mpciter>=100)%Not used, but was introduced for parameter
            Sub=-1              variation while running simulation
            else
            Sub=Lnew-LL(end)
            end
            Nero=Res1(end,1)
            Const=Res1(end,2)
            Const2=Res1(end,3)
            if (Nero>0.000001)||(Const>0)||(Const2>0)||(Sub>0)
            YZW=1;

            else
            YZW=-1;

            end
        else
            YZW=-1;
            Sub=-1;
            Nero=Nero0;
            Const=Res1(end,2);
            Const2=Res1(end,3);

        end



        % Step (3) of the NMPC algorithm:
        %   Apply control to process
        [tmeasure,
xmeasure,Pmeasure,Adismeasure,Neromeasure,Constmeasure,Constmeasure2,factor,Y
YZ,W] =
applyControl(system,system3,system4,system5,system6,proceso2,computeu,B0,Nero
,Nero0,Const,Const0,Const20,Const2,P0,S,x1b,mpciter,YZW,YYZ0,factor0,Sub,Sub0
,u0,N, T, t0, x0, u_new, ...
            atol_ode_real, rtol_ode_real, type);


        U21=[U;W];


            if (mpciter==1)
```

```
    t = TTT;
    x = XXX;
    u = UUU;
    Res = Res1
    LL=LL1
    BB=BB1;
    U=U21
    elseif (YZW==1)||(factor==10)
    mpciter=mpciter-1;
    t = TTT(1:end-1,:);
    x = XXX(1:end-1,:);
    u = UUU(1:end-1,:);
    Res = Res1(1:end-1,:)
    LL=LL1(1:end-1,:)
    BB=BB1(1:end-1,:);
    U=U21(1:end-1,:)
    JJJ=mpciter+1
    else
    t = TTT;
    x = XXX;
    u = UUU;
    Res = Res1
    LL=LL1
    BB=BB1;
    U=U21
    JJJ=0
    end

Ameasure=LL(end)*(Adismeasure/Adismeasure)

Neromeasure
Constmeasure
Constmeasure2

u0 = shiftHorizon(u_new,N,Pmeasure);

if(mpciter>1)
```

```
        if ((Nero<=0)&&(Const<0)&&(Const2<0)&&(S==0)&&(Sub>0))
        mpciter=mpciterations;
        end %Introduced to stop simulation when the Lyapunov Robust Stability
        end  constraint is not a good approximation

        mpciter = mpciter+1;

        clear TTT;
        clear UUU;
        clear XXX;

    end
end


function [t0, x0, P0, A0, Nero0, Const0, Const20, factor0,YYZ0,Sub0] =
measureInitialValue ( tmeasure, xmeasure,
Pmeasure,Ameasure,Neromeasure,Constmeasure,Constmeasure2,factor,YYZ,Sub)
    t0 = tmeasure;
    x0 = xmeasure;
    P0 = Pmeasure;
    A0 = Ameasure;
    Nero0=Neromeasure;
    Const0=Constmeasure;
    Const20=Constmeasure2;
    factor0=factor;
    YYZ0=YYZ;
    Sub0=Sub;
end


function y = lyapunov
(system,system3,system4,system5,obtencionP,mpciter,Nero0,Const0,Const20,P0,fa
ctor0,YYZ0,Sub0, N, T, t0, x0, u, ...
                                    atol_ode_sim, rtol_ode_sim, type)


    x = zeros(N+1, length(x0));
    x=
computeOpenloopSolution(system,system3,system4,system5,mpciter,Nero0,Const0,C
onst20,factor0,YYZ0,Sub0, N, T, t0, x0, u, ...
```

```
                                atol_ode_sim, rtol_ode_sim, type);

    y = obtencionP(t0,x,u,N,Nero0,P0,Const20,Const0,Sub0);

end

function [tapplied, xapplied, Papplied, Aapplied, Neroapplied, Constapplied,
Constapplied2,factor,YYZ,W]=
applyControl(system,system3,system4,system5,system6,proceso2,computeu,B0,Nero
,Nero0,Const,Const0,Const20,Const2,P0,S,x1b,mpciter,YZW,YYZ0,factor0,Sub,Sub0
,u0,N, T, t0, x0, u,atol_ode_real, rtol_ode_real, type)

                              if (mpciter<= 0)
    if (YZW>0)||(u(4,1)==2.4310)
    AAA=abs(Const2);
    else
    AAA=Const2;
    end



    xa = system4(t0, x0, u(:,1), T, mpciter,AAA,YYZ0);

    if (YZW>0)
    SS=x0(1,1:4);
    UUU=computeu(N, T, t0, x1b, u,mpciter,xa,S);
    W=UUU(1,:)
    YYZ=0;
    else
    UUU=computeu(N, T, t0, x1b, u,mpciter,xa,S);
    W=UUU(1,:)
    xapplied1 = dynamic(system,system3,system5, T, t0, xa, S*u(:,1), ...
                        atol_ode_real, rtol_ode_real, type);
    SS=xapplied1(1,1:4)+xa(1,5:8);
    YYZ=0;
    end

   if (YZW<0)
    if (max(W(:,1))>1.00001)
```

```
        factor=10
    else
        factor=0
    end
   else
      factor=0
   end



    xapplied = [SS,xa(1,5:11)]
    tapplied = t0+T;

[Papplied,Constapplied2]=system6(u,N,P0,Nero,Nero0,Const0,Const,Const2,Const2
0,YZW,factor,Sub,factor0,Sub0,u0);
    Aapplied=B0;
    Neroapplied=Nero;
    Constapplied=Const;
    Papplied


      %Used for parameter variation
                           else
                               baldano=u(4,1)-2.4310
    if (baldano<=0.000001)||(YZW>0)
    AAA=abs(Const2)
    else
    AAA=Const2
    end



    xa = system4(t0, x0, u(:,1), T, mpciter,AAA,YYZ0)


    if (YZW>0)
    SS=x0(1,1:4);
    UUU=computeu(N, T, t0, x1b, u,mpciter,xa,S);
    W=UUU(1,:)
    %if (mpciter>4)
    %xver=dynamic2(proceso2, T, t0, ...
```

```
            %x0(1,1:4), UUU(1,:),  atol_ode_real, rtol_ode_real);

    %YYZ=(abs((xver-SS)))*ones(4,1)
    %else
    YYZ=0;
    %end
    xapplied = [SS,xa(1,5:11)]
    else
    UUU=computeu(N, T, t0, x1b, u,mpciter,xa,S);
    W=UUU(1,:)
    xapplied1 = dynamic(system,system3,system5, T, t0, xa, S*u(:,1), ...
                        atol_ode_real, rtol_ode_real, type);
    SS=xapplied1(1,1:4)+xa(1,5:8);

    xver=dynamic2(proceso2, T, t0, ...
             x0(1,1:4), UUU(1,:),  atol_ode_real, rtol_ode_real);

    YYZ=(abs((xver-SS)))*ones(4,1)
    xapplied = [xver,xa(1,5:11)]
    end

    if (YZW<0)
    if (max(W(:,1))>1.00001)
        factor=10
    else
        factor=0
    end
   else
     factor=0
    end

    tapplied = t0+T;

[Papplied,Constapplied2]=system6(u,N,P0,Nero,Nero0,Const0,Const,Const2,Const2
0,YZW,factor,Sub,factor0,Sub0,u0);
    Aapplied=B0;
    Neroapplied=Nero;
```

```
    Constapplied=Const;
    Papplied




                            end

end


function u0 = shiftHorizon(u,N,P0)
    A=zeros(10,N-1);
    B=[P0',A];
    u0 = [zeros(2,N);0.999999*ones(1,N);2.4310*ones(1,N);B;ones(1,N)];
end


function [u, V, exitflag, output] = solveOptimalControlProblem ...
    (runningcosts, terminalcosts, constraints, terminalconstraints, ...
    linearconstraints, system,
avconstraint,system3,system4,system5,constraintstotales,constraintstotales2,m
pciter,Nero0,Const0,Const20, N,P0,A0,factor0,YYZ0,Sub0, t0, x0, u0, T, ...
    atol_ode_sim, rtol_ode_sim, tol_opt, options, type)
    x = zeros(N+1, length(x0));
    x=
computeOpenloopSolution(system,system3,system4,system5,mpciter,Nero0,Const0,C
onst20,factor0,YYZ0,Sub0, N, T, t0, x0, u0, ...
                                atol_ode_sim, rtol_ode_sim, type);



    % Set control and linear bounds
    A = [];
    b = [];
    Aeq = [];
    beq = [];
    lb = [];
    ub = [];
    for k=1:N
```

```
        [Anew, bnew, Aeqnew, beqnew, lbnew, ubnew] = ...
               linearconstraints(t0+k*T,x(k,:),u0(:,k));
        A = blkdiag(A,Anew);
        b = [b, bnew];
        Aeq = blkdiag(Aeq,Aeqnew);
        beq = [beq, beqnew];
        lb = [lb, lbnew];
        ub = [ub, ubnew];
    end


    % Solve optimization problem
    [u, V, exitflag, output] = fmincon(@(u) real(costfunction(runningcosts,
...
        terminalcosts,
system,system3,system4,system5,mpciter,Nero0,Const0,Const20,factor0,YYZ0,Sub0
, N, T, t0, x0, ...
        u, atol_ode_sim, rtol_ode_sim, type)), u0, A, b, Aeq, beq, lb, ...
        ub, @(u) nonlinearconstraints(constraints, terminalconstraints, ...
        system,
avconstraint,system3,system4,system5,constraintstotales,constraintstotales2,P
0,A0,mpciter,Nero0,Const0,Const20,factor0,YYZ0,Sub0, N, T, t0, x0, u, ...
        atol_ode_sim, rtol_ode_sim, type), options);



end


function cost = costfunction(runningcosts, terminalcosts, system, ...

system3,system4,system5,mpciter,Nero0,Const0,Const20,factor0,YYZ0,Sub0,N, T,
t0, x0, u, ...
                    atol_ode_sim, rtol_ode_sim, type)
    cost = 0;
    x = zeros(N+1, length(x0));
    x=
computeOpenloopSolution(system,system3,system4,system5,mpciter,Nero0,Const0,C
onst20,factor0,YYZ0,Sub0, N, T, t0, x0, u, ...
                                atol_ode_sim, rtol_ode_sim, type);
    for k=1:N
        cost = cost+runningcosts(t0+k*T, x(k,:), u(:,k),N,mpciter);
    end
```

```
    cost = cost+terminalcosts(t0+(N+1)*T, x(N+1,:));
end


function [c,ceq] = nonlinearconstraints(constraints, ...
    terminalconstraints, system, ...

avconstraint,system3,system4,system5,constraintstotales,constraintstotales2,P
0,A0,mpciter,Nero0,Const0,Const20,factor0,YYZ0,Sub0,N, T, t0, x0, u,
atol_ode_sim, rtol_ode_sim, type)
    x = zeros(N+1, length(x0));
    x=
computeOpenloopSolution(system,system3,system4,system5,mpciter,Nero0,Const0,C
onst20,factor0,YYZ0,Sub0, N, T, t0, x0, u, ...
                                atol_ode_sim, rtol_ode_sim, type);


   if (Const20<0)
       if (Nero0<=0.000001)
           if (Const0>0)||(factor0==10)||(Sub0>0)
              S=0;


           else
              S=1;


           end
       else
        S=1;


       end
   else
       S=1;


   end


    c = [];
    ceq = [];
    for k=1:N
        [cnew, ceqnew] = constraints(t0,x(k,:),S*u(1:2,k),mpciter);
        c = [c cnew];
```

```
        ceq = [ceq ceqnew];
    end
    [cnew1 ceqnew1] = avconstraint(t0,x,u,N,T,mpciter);
    c = [c cnew1];
    ceq = [ceq ceqnew1];
    if (mpciter==1)
    [cnew2 ceqnew2] = constraintstotales(t0,x,u,N,T,S);
    c = [c cnew2];
    ceq = [ceq ceqnew2];
    else
    [cnew2 ceqnew2] =
constraintstotales2(t0,x,u,N,P0,A0,T,S,Nero0,Const20,Const0,Sub0);
    c = [c cnew2];
    ceq = [ceq ceqnew2];
    end
    [cnew3, ceqnew3] = terminalconstraints(t0+k*T,x,u,N,mpciter);
    c11 = [c cnew3];
    ceq11 = [ceq ceqnew3];
    c1 = real(c11);
    ceq1 = real (ceq11);
    c = c1;
    ceq = ceq1;
end


function x=
computeOpenloopSolution(system,system3,system4,system5,mpciter,Nero0,Const0,C
onst20,factor0,YYZ0,Sub0, N, T, t0, x0, u, ...
                                    atol_ode_sim, rtol_ode_sim, type)
   global JJJ


   if (mpciter>1)
       if (Const20==-2)&&(JJJ==mpciter)
       Value=abs(Const20);
       else
       Value=Const20;
       end
   else
       Value=Const20;
```

```
    end

     x(1,:) = system4(t0, x0, u, T, mpciter,Value,YYZ0);



    if (Const20<0)
        if (Nero0<=0.000001)
            if (Const0>0)||(factor0==10)||(Sub0>0)
               S=0;

            else
               S=1;

            end
        else
         S=1;

        end
    else
        S=1;

    end



     for k=1:N
         x(k+1,:) = dynamic(system,system3,system5, T, t0, x(k,:), S*u(1:2,k),
...
                            atol_ode_sim, rtol_ode_sim, type);
     end



end
```

```
function [x, t_intermediate, x_intermediate] =
dynamic(system,system3,system5, T, t0, ...
             x0, u, atol_ode, rtol_ode, type)
    if ( strcmp(type, 'difference equation') )
        x = system(t0, x0, u, T);
        x_intermediate = [x0; x];
        t_intermediate = [t0, t0+T];
    elseif ( strcmp(type, 'differential equation') )
        xs1=system5(x0(1,5:8),u);
        options = odeset('AbsTol', atol_ode, 'RelTol', rtol_ode);
        [t_intermediate,x_intermediate] = ode45(system, ...
            [t0, t0+T], x0, options, u);
        x2 = x_intermediate(size(x_intermediate,1),:);
        x1=system3(t0, x0, u, T);

        x = [x2(1,1:4),xs1,x1];
    end
end


function [x, t_intermediate, x_intermediate] = dynamic2(proceso2, T, t0, ...
             x0, u, atol_ode, rtol_ode)

        options = odeset('AbsTol', atol_ode, 'RelTol', rtol_ode);

        [t_intermediate,x_intermediate] = ode45(proceso2, ...
            [t0, t0+T], x0, options, u);

        x2 = x_intermediate(size(x_intermediate,1),:);

        x = x2;

end
```

Steady State Model and generation of deviation variables.

```
function y = inicializa(t, x0, u, T,mpciter,Const20,YYZ0)
```

```
global s1

US = [u(3,1),u(4,1)];

if (YYZ0*0<0.00018) % Check for parameter adaptation
s1=x0(1,11);
else
s1=u(15,1);
end



function y = valores(x0, u)

options = optimset('MaxIter',600,'Display','off','Jacobian','on');
    function [H,J] = steady(z)
    s2=0.4;

    H = [u(1)-z(1)-s1*z(1)*z(2);
        u(2)-z(2)-s1*z(1)*z(2)-s2*z(2)*z(3);
        -z(3)+s1*z(1)*z(2)-s2*z(2)*z(3);
        -z(4)+s2*z(2)*z(3)];

    J = [ -s1*z(2)-1, -s1*z(1), 0,  0;
         -s1*z(2), -1-s1*z(1)-s2*z(3), -s2*z(2), 0;
         s1*z(2), s1*z(1)-s2*z(3),-1-s2*z(2), 0;
         0, s2*z(3),s2*z(2), -1];

    end

X = fsolve(@steady,[x0(1,5);x0(1,6);x0(1,7);x0(1,8)],options);


y = X';
end
```

```
    if (Const20<0)

    L = valores(x0, US);

    y5=L(1);
    y6=L(2);
    y7=L(3);
    y8=L(4);

    y9=US(1);
    y10=US(2);
    y11=s1;


    xsprocess = [y5,y6,y7,y8];
    xprocess = x0(1,1:4);
    dev = xprocess-xsprocess;

    y = [dev,y5,y6,y7,y8,y9,y10,y11];


    else

    y11=s1;
    xsprocess = x0(1,5:8);
    xprocess = x0(1,1:4);
    dev = xprocess-xsprocess;

    y = [dev,x0(1,5:8),x0(1,9:10),y11];

    end


end
```

Dynamic deviation model

```
function y = contsystem_ct(t, x, u, T)

    s1=x(11);
    s2=0.4;

    a = x(1:4);

    ud = [u(1)*(a(1)+a(2)+a(3)+a(4));u(2)*(a(1)+a(2)+a(3)+a(4))];

    %SS System
    dyy = zeros(7,1);
    y2= dyy;

    %Deviation System
    dy = zeros(4,1);
    dy(1) = (ud(1)+x(9))-(x(1)+x(5))-s1*(x(1)+x(5))*(x(2)+x(6));
    dy(2) = (ud(2)+x(10))-(x(2)+x(6))-s1*(x(1)+x(5))*(x(2)+x(6))-
s2*(x(2)+x(6))*(x(3)+x(7));
    dy(3) = -(x(3)+x(7))+s1*(x(1)+x(5))*(x(2)+x(6))-
s2*(x(2)+x(6))*(x(3)+x(7));
    dy(4) = -(x(4)+x(8))+s2*(x(2)+x(6))*(x(3)+x(7));
    y1= [dy(1);dy(2);dy(3);dy(4)];
    yt = [y1;y2];
    y = real(yt);
end
```

True model when adapting parameters

```
function y = proceso2(t, x, u, T)

    %Factores de dimensionless mass balance
    s1=1.02;
    s2=0.4;
```

```
    %System
    dy = zeros(4,1);
    dy(1) = u(1)-x(1)-s1*x(1)*x(2);
    dy(2) = u(2)-x(2)-s1*x(1)*x(2)-s2*x(2)*x(3);
    dy(3) = -x(3)+s1*x(1)*x(2)-s2*x(2)*x(3);
    dy(4) = -x(4)+s2*x(2)*x(3);
    y1= [dy(1);dy(2);dy(3);dy(4)];

    y = real(y1);
end
```

Function that fixes the steady states

```
function y = valoresxs(x, u)

    y = x;
end
```

Function that fixes the manipulated inputs at steady state

```
function y = evaluaciones(t, x, u, T)


    p=x(1,9:11);


    y = p;
end
```

Upper and lower bound constraints. Here the bounds on the parameter must be set

```
function [c,ceq] = constraints(t, x, u, mpciter)

    a = x(1:4);
    ud = [u(1)*(a(1)+a(2)+a(3)+a(4));u(2)*(a(1)+a(2)+a(3)+a(4))];
```

```
     c(1)=x(9)-1;
     c(2)= -1*x(9);
     c(3) = -1*x(10);
     c(4)=x(10)-10;
     c(5)=(ud(1)+x(9))-0.99999999;
     c(6)= -1*(ud(1)+x(9));
     c(7) = -1*(ud(2)+x(10));
     c(8)=(ud(2)+x(10))-10;
     c(9)=x(11)-1.0299;
     c(10)=-x(11)+1;

    ceq = [];
end
```

Average Constraints

```
function [c,ceq] = avconstraint(t,x,u,N,T,mpciter)

 c =[];
 ceq = [];

end
```

Special Constraints at initialization. We applied these constraints since at initialization it is not necessary to satisfied the set point constraint

```
function [c,ceq] = constraintstotales(t, x, u,N,T,S)
    alfa = 0.00000000001;
    QA=alfa*(ones(4,1));

   function  ZZZ = funcionLya(P01,x,N)

    PP = [P01(1) P01(2) P01(3) P01(4); P01(2) P01(5) P01(6) P01(7); P01(3)
P01(6) P01(8) P01(9); P01(4) P01(7) P01(9) P01(10)];

    for k=1:N
```

```
      Z(k,1)= (x(k,1:4)*PP)*x(k,1:4)';

    end

    ZZZ=ones(1,N)*Z(:,1);
    end

    Ai=[u(5,1) u(6,1) u(7,1) u(8,1) u(9,1) u(10,1) u(11,1) u(12,1) u(13,1)
u(14,1)];
    Pi=[u(5,1) u(6,1) u(7,1) u(8,1); u(6,1) u(9,1) u(10,1) u(11,1); u(7,1)
u(10,1) u(12,1) u(13,1); u(8,1) u(11,1) u(13,1) u(14,1)];



    Lyk = funcionLya(Ai,x,N);



    con3=-1*(real(eig(Pi)))+QA;

    con1= 0;

    [teta,beta]=evaluacionq(Ai,x,u,T,S,N);
    con2=(-1*beta-Lyk)+alfa;



    c=[con1;con2;con3]';



    ceq = [];
end



function [e,y]=evaluacionq(P01,x,u,T,S,N)
u01=[0.3;0.3;0.3];
options = optimset('Display','notify',...
                'TolFun', 1e-6,...
```

```
                'MaxFunEvals', 3001,...
                'MaxIter', 4000,...
                'Algorithm', 'sqp',...
                'AlwaysHonorConstraints', 'bounds',...
                'FinDiffType', 'forward',...
                'HessFcn', [],...
                'Hessian', 'bfgs',...
                'HessMult', [],...
                'InitBarrierParam', 0.1,...
                'InitTrustRegionRadius', sqrt(size(u01,1)*size(u01,2)),...
                'MaxProjCGIter', 2*size(u01,1)*size(u01,2),...
                'ObjectiveLimit', -1e20,...
                'ScaleProblem', 'obj-and-constr',...
                'SubproblemAlgorithm', 'cg',...
                'TolProjCG', 1e-2,...
                'TolProjCGAbs', 1e-10);


A=[];b=[];
Aeq=[1,1,1];beq=1;
lb=[0;0;0];ub=[1;1;1];
nonlcon=[];



function h=mifuncion(P01,x,u,T,S,N,U1)
[AclT,Acl1T,Acl2T] = funcionP2(P01,x,u,T,S,N);

ACL = AclT*U1(1)+Acl1T*U1(2)+Acl2T*U1(3);

h=-1*ACL;

end
```

```
[e,y] = fmincon(@(U1)
mifuncion(P01,x,u,T,S,N,U1),u01,A,b,Aeq,beq,lb,ub,nonlcon,options);
end


function [AclT,Acl1T,Acl2T] = funcionP2(P01,x,u,T,S,N)

    %Definicion de parametros y variables
    s1=x(1,11);
    s2=0.4;
    gama=T;

    %Transient
    %Minimos
    x10 = x(5);
    x20 = x(6);
    x30 = x(7);

    %K1=[u(1) u(1) u(1) u(1);u(2) u(2) u(2) u(2)];

    A = [((-s1*x20-1)*T+1) (-s1*x10)*T (0) (0);(-s1*x20)*T ((-1-s1*x10-
s2*x30)*T+1) (-s2*x20)*T (0);(s1*x20)*T (s1*x10-s2*x30)*T ((-1-s2*x20)*T+1)
(0);(0) (s2*x30)*T (s2*x20)*T (-1*T+1)];
    I1 = 0.01*ones(4,4);
    A1 = A+I1;
    A2 = A-I1;
    B = [1 0;0 1;0 0;0 0];
    PP = [P01(1) P01(2) P01(3) P01(4); P01(2) P01(5) P01(6) P01(7); P01(3)
P01(6) P01(8) P01(9); P01(4) P01(7) P01(9) P01(10)];

    for k=1:N

        Acl(k,1)=
((((A+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'))'*P
P)*(((A+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'));
        Acl1(k,1)=
((((A1+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'))'*
PP)*(((A1+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)')
);
        Acl2(k,1)=
((((A2+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'))'*
```

```
PP)*(((A2+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)')
);

    end



    AclT = ones(1,N)*Acl(:,1);
    Acl1T = ones(1,N)*Acl1(:,1);
    Acl2T = ones(1,N)*Acl2(:,1);




end
```

Special constraints for iter>1. These are the same as constraint at initialization, however, the set point constraint is considered and a logic memory is introduced to satisfied feasibility

```
function [c,ceq] = constraintstotales2(t, x,
u,N,P0,A0,T,S,Nero,Const20,Const,Sub)
    alfa = 0.00000000001;
    QA=alfa*(ones(4,1));


    function  ZZZ = funcionLya(P01,x,N)


    PP = [P01(1) P01(2) P01(3) P01(4); P01(2) P01(5) P01(6) P01(7); P01(3)
P01(6) P01(8) P01(9); P01(4) P01(7) P01(9) P01(10)];


    for k=1:N


      Z(k,1)= (x(k,1:4)*PP)*x(k,1:4)';


    end


    ZZZ=ones(1,N)*Z(:,1);
    end
```

```
    Ai=[u(5,1) u(6,1) u(7,1) u(8,1) u(9,1) u(10,1) u(11,1) u(12,1) u(13,1)
u(14,1)];

    Pi=[u(5,1) u(6,1) u(7,1) u(8,1); u(6,1) u(9,1) u(10,1) u(11,1); u(7,1)
u(10,1) u(12,1) u(13,1); u(8,1) u(11,1) u(13,1) u(14,1)];

    PPi = [P0(1) P0(2) P0(3) P0(4); P0(2) P0(5) P0(6) P0(7); P0(3) P0(6)
P0(8) P0(9); P0(4) P0(7) P0(9) P0(10)];


    if (Const20<0)
        if (Nero<=0.000001)
            if (Const<0)
                if(Sub<0)
                    %eig(P)>0
                    %Set point = xk'P(k-1)xk-x(k-1)'P(k-1)x(k-1)<0
                    %P = xk'P(k)xk-x(k)'P(k-1)x(k)<=0
                    %Robust = max(xk,Acl,P(k))-x(k)'P(k)x(k)<0


            con3=-1*(real(eig(Pi)))+QA;
            Lyk = funcionLya(Ai,x,N);
            Lykb = funcionLya(P0,x,N);

            con1= Lyk-Lykb;

            con5=Lykb-A0+alfa;



            [teta,beta]=evaluacionq(Ai,x,u,T,S,N);
            con2=(-1*beta-Lyk)+alfa;

            c=[con1;con2;con5;con3]';



            ceq = [];
                else

        con3=-1*(real(eig(PPi)))+QA;
        Lyk = funcionLya(P0,x,N);
```

```
Lykb = funcionLya(P0,x,N);

con1= Lyk-Lykb;

con5=-2;


[teta,beta]=evaluacionq(P0,x,u,T,S,N);
con2=(-1*beta-Lyk)+alfa;

c=[con1;con2;con5;con3]';


ceq = [];
        end
    else
con3=-1*(real(eig(PPi)))+QA;
Lyk = funcionLya(P0,x,N);
Lykb = funcionLya(P0,x,N);

con1= Lyk-Lykb;

con5=-2;


[teta,beta]=evaluacionq(P0,x,u,T,S,N);
con2=(-1*beta-Lyk)+alfa;

c=[con1;con2;con5;con3]';


ceq = [];

    end
else
```

```
    con3=-1*(real(eig(PPi)))+QA;
    Lyk = funcionLya(P0,x,N);
    Lykb = funcionLya(P0,x,N);

    con1= Lyk-Lykb;

    con5=-2;


    [teta,beta]=evaluacionq(P0,x,u,T,S,N);
    con2=(-1*beta-Lyk)+alfa;

    c=[con1;con2;con5;con3]';


    ceq = [];

    end
else

    con3=-1*(real(eig(Pi)))+QA;
    Lyk = funcionLya(Ai,x,N);
    Lykb = funcionLya(P0,x,N);

    con1= Lyk-Lykb;

    con5=-2;


    [teta,beta]=evaluacionq(Ai,x,u,T,S,N);
    con2=(-1*beta-Lyk)+alfa;

    c=[con1;con2;con5;con3]';
```

```
        ceq = [];

    end

end


function [e,y]=evaluacionq(P01,x,u,T,S,N)
u01=[0.3;0.3;0.3];
options = optimset('Display','notify',...
                'TolFun', 1e-6,...
                'MaxFunEvals', 3001,...
                'MaxIter', 4000,...
                'Algorithm', 'sqp',...
                'AlwaysHonorConstraints', 'bounds',...
                'FinDiffType', 'forward',...
                'HessFcn', [],...
                'Hessian', 'bfgs',...
                'HessMult', [],...
                'InitBarrierParam', 0.1,...
                'InitTrustRegionRadius', sqrt(size(u01,1)*size(u01,2)),...
                'MaxProjCGIter', 2*size(u01,1)*size(u01,2),...
                'ObjectiveLimit', -1e20,...
                'ScaleProblem', 'obj-and-constr',...
                'SubproblemAlgorithm', 'cg',...
                'TolProjCG', 1e-2,...
                'TolProjCGAbs', 1e-10);


A=[];b=[];
Aeq=[1,1,1];beq=1;
lb=[0;0;0];ub=[1;1;1];
nonlcon=[];
```

```
function h=mifuncion(P01,x,u,T,S,N,U1)
[AclT,Acl1T,Acl2T] = funcionP2(P01,x,u,T,S,N);

ACL = AclT*U1(1)+Acl1T*U1(2)+Acl2T*U1(3);

h=-1*ACL;

end
```

```
[e,y] = fmincon(@(U1)
mifuncion(P01,x,u,T,S,N,U1),u01,A,b,Aeq,beq,lb,ub,nonlcon,options);
end
```

```
function [AclT,Acl1T,Acl2T] = funcionP2(P01,x,u,T,S,N)

    %Definicion de parametros y variables
    s1=x(1,11);
    s2=0.4;
    gama=T;

    %Transient
    %Minimos
    x10 = x(5);
    x20 = x(6);
    x30 = x(7);

    %K1=[u(1) u(1) u(1) u(1);u(2) u(2) u(2) u(2)];

    A = [((-s1*x20-1)*T+1) (-s1*x10)*T (0) (0);(-s1*x20)*T ((-1-s1*x10-
s2*x30)*T+1) (-s2*x20)*T (0);(s1*x20)*T (s1*x10-s2*x30)*T ((-1-s2*x20)*T+1)
(0);(0) (s2*x30)*T (s2*x20)*T (-1*T+1)];
    I1 = 0.01*ones(4,4);
    A1 = A+I1;
    A2 = A-I1;
    B = [1 0;0 1;0 0;0 0];
```

```
    PP = [P01(1) P01(2) P01(3) P01(4); P01(2) P01(5) P01(6) P01(7); P01(3)
P01(6) P01(8) P01(9); P01(4) P01(7) P01(9) P01(10)];


    for k=1:N


        Acl(k,1)=
((((A+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'))'*P
P)*(((A+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'));
        Acl1(k,1)=
((((A1+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'))'*
PP)*(((A1+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)')
);
        Acl2(k,1)=
((((A2+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)'))'*
PP)*(((A2+(B*[u(1,k)*gama*S*ones(1,4);u(2,k)*gama*S*ones(1,4)])))*(x(k,1:4)')
);


    end



    AclT = ones(1,N)*Acl(:,1);
    Acl1T = ones(1,N)*Acl1(:,1);
    Acl2T = ones(1,N)*Acl2(:,1);




end
```

Linear Constraints (Used constraints.mat to set upper and lower bounds)

```
function [A, b, Aeq, beq, lb, ub] = linearconstraints(t, x, u)
    A   = [];
    b   = [];
    Aeq = [];
    beq = [];
    ub  = [];
    lb = [];


end
```

Terminal Constraints

```
function [c,ceq] = terminalconstraints(t,x,u,N,mpciter)


    c   = [];
    ceq =[];



end
```

Stage Cost

```
function cost = runningcosts(t, x, u, N, mpciter)


Q=45;
cost = -(x(3)+Q*x(7));


end
```

Terminal Cost

```
function cost = terminalcosts(t, x)
    cost = 0.0;
end
```

Memory for the controller

```
function [y,yy] =
evalP(u,N,P0,Nero,Nero0,Const0,Const,Const2,Const20,YZW,factor,Sub,factor0,Su
b0,u0)


    if (Const2==-2)
    yy=Const2;
    elseif (YZW>0)
    yy=abs(Const2);
```

```
    else
    yy=Const2;
    end

    if (Const2<0)
        if (Nero<=0.000001)
            Delta=(u(5:14,1)'-u0(5:14,1)')*ones(10,1);
            if (Const>0)||(factor==10)||(Delta==0)||(Sub>0)
                y=P0;
            else
                y=u(5:14,1)';
            end
        else
        y=P0;
        end
    else
    y=P0;
    end


end
```

Computation of Lyapunov

```
function y = obtencionP(t,x,u,N,Nero0,Pm,Const20,Const0,Sub)

    function  ZZZ = funcionLya(P01,x,N)

    PP = [P01(1) P01(2) P01(3) P01(4); P01(2) P01(5) P01(6) P01(7); P01(3)
P01(6) P01(8) P01(9); P01(4) P01(7) P01(9) P01(10)];

    for k=1:N

      Z(k,1)= (x(k,1:4)*PP)*x(k,1:4)';

    end
```

```
    ZZZ=ones(1,N)*Z(:,1);
    end

    Ai=[u(5,1) u(6,1) u(7,1) u(8,1) u(9,1) u(10,1) u(11,1) u(12,1) u(13,1)
u(14,1)];



    if (Const20<0)
        if (Nero0<=0.000001)
            if (Const0<0)
                if (Sub<0)

        y = funcionLya(Ai,x,N);

                else
        y = funcionLya(Pm,x,N);

                end

            else
        y = funcionLya(Pm,x,N);
            end
        else
        y = funcionLya(Pm,x,N);

        end
    else
    y = funcionLya(Ai,x,N);

    end


end
```

Computation of Ameasure

```
function A= obtencionA(t,x,u,N,Nero0,Pm,Const20,Const0,Sub,A0)

A=1;

end
```